WWW.HACKERJOURNAL.IT

2€ NO PUBBLICITÀ SOLO INFORMAZIONI E ARTICOLI

HACKER JOURNAL

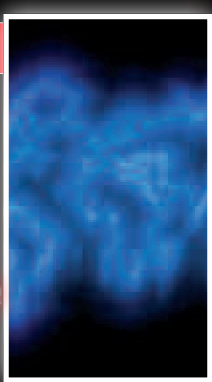
N° 201



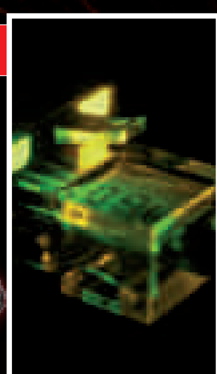
CORSO DI C
PROGRAMMAZIONE IN C
SECONDA PARTE

PREVENIRE LO SNIFFING

L'ALGORITMO DI CIFRATURA RSA



MOBILE
› GPS TRACKING CON IL CELLULARE



ATTUALITÀ
› P2P UNA NORMATIVA "NEBULOSA"

PROGRAMMAZIONE
› DOWNLOAD A PROVA DI BOMBA

QUATTORD. ANNO 10 - N° 201 - 13 MAGGIO/26 MAGGIO 2010 - € 2,00

EDITORIALE

UN CORSO COI FIOCCHI

In questo numero abbiamo un menu particolarmente ricco, tuttavia volevo invitarvi a seguire il corso di programmazione in C, giunto alla seconda puntata, perché a mio avviso è davvero meritevole di essere letto e conservato come una sorta di guida a dispense.
Come molti lettori e frequentatori del forum già sapranno, l'idea di realizzare il corso è maturata proprio dal confronto coi lettori e con gli utenti del sito. C'era richiesta e attesa. Ritengo che proprio le aspettative non siano andate deluse perché il corso di programmazione in C si sta evolvendo in modo davvero efficace.
Il problema, quando si approccia un argomento così vasto, è di risultare o troppo superficiali o troppo tecnici, oppure entrambe le cose.
Giovanni e Fabio, gli autori e ideatori del corso, stanno invece riuscendo a mantenersi in una perfetta situazione di equilibrio, condensando nel poco spazio disponibile, se non tutto, molto di quello che gli utenti si aspetterebbero di trovare.
Sono convinto che si tratti davvero di un buon lavoro, voi che ne pensate?

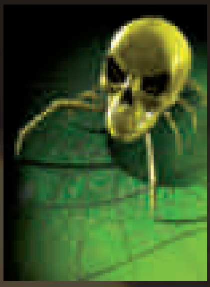
Altair

laboratorio@hackerjournal.it
Questo indirizzo è stato creato per inviare articoli, codici, spunti e idee. E' quindi proprio una sorta di "incubatore di idee".

posta@hackerjournal.it
E' l'account creato per l'omonima rubrica che è ricomparsa nelle pagine della rivista. A questo indirizzo dovete inviare tutte le mail che volete vengano pubblicate su HJ.

redazione@hackerjournal.it
Questo è l'indirizzo canonico. Quello con cui potete avere un filo diretto, sempre, con la redazione, per qualsiasi motivo che non rientri nelle due precedenti categorie di posta.

Sommario



Copertina:
Daniela Festa
ldfesta@libero.it

Anno 10 - N.201
13 maggio / 26 maggio 2010

Editore (sede legale)
WLF Publishing S.r.l.
Socio Unico Medi & Son S.r.l.
via Donatello 71 - 00196 Roma
Fax 063214606

Realizzazione editoriale
Progetti e promozioni Srl
redazione@progettiepromozioni.com

Printing
Grafiche Mazzucchelli S.p.a - Seriate (BG)

Distributore
M-DIS Distributore SPA
via Cazzaniga 2 - 20123 Milano

Hacker Journal
Pubblicazione quattordicinale registrata al Tribunale di Milano il 27/10/03 con il numero 601.
Una copia: 2,00 euro

Direttore Responsabile
Teresa Carsaniga
redazione@hackerjournal.it

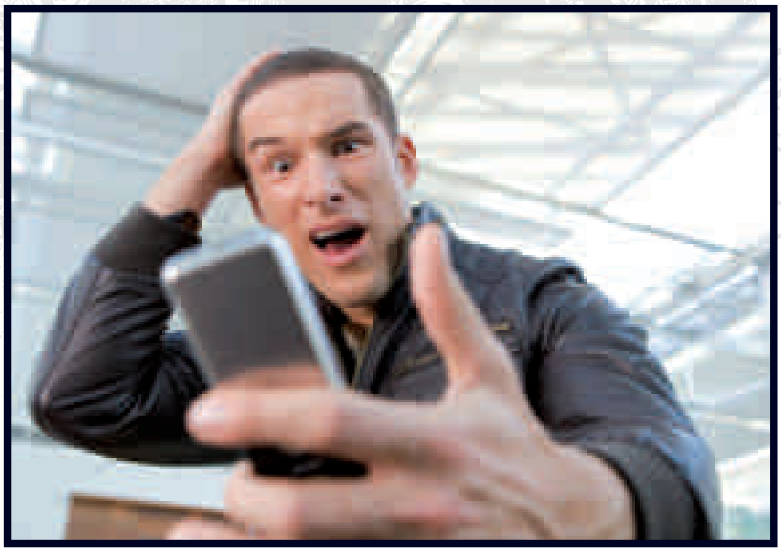
FUTURO E PROSPETTIVE DELL' OPEN SOURCE



Uno degli aspetti decisamente appaganti di lavorare nella redazione di una rivista come Hacker Journal è rappresentato dal contatto diretto coi lettori. Questo rapporto decisamente confidenziale permette di avere riscontri pressoché subitanei rispetto ad articoli pubblicati sulla rivista.
E' un po' quello che è accaduto per l'articolo relativo a Richard Stallman, pubblicato sul numero 199, che ha dato vita ad un interessante confronto di idee in parte ripreso anche nella consueta rubrica della posta.
Ci sembra di intuire che il tema dei software a "sorgenti aperti" sia particolarmente popolare tra i lettori che ne esaltano le doti di affidabilità e sicurezza.
Però tutto questo entusiasmo forse stride un po' con la situazione reale. Infatti se sul fronte server si è assistito davvero a un'affermazione del software open source e delle soluzioni Linux oriented, sul fronte del software per uso privato o lavorativo la sensazione è che l'avanzata del software open source sia più lenta, più macchinosa. Il software commerciale sembra non recedere dalla posizioni di predominio.
Obiettivamente se dieci anni fa ci avessero chiesto una previsione sullo stato della diffusione del software open source nel 2010, avremmo, con ogni probabilità, tratteggiato una situazione più rosea.
Certo rimangono casi di assoluta eccellenza. Firefox è con Exploer il browser più diffuso, Gimp e Open Office sono molto noti e utilizzati come suite di office e programma di fotoritocco. Ma poi?
La sensazione è che forse sul fronte dello sviluppo occorre essere "più furbi" per cercare di scardinare le regole di un mercato che sembra immutabile.
Un esempio?
Blender è davvero un ottimo programma di 3D, capace di competere con blasonati software commerciali. Ha tutto quello che serve: modellatore, animazione e effetti particellari. Il problema è che ha un'interfaccia grafica "difficile", troppo diversa da quella a cui sono abituati gli utenti che utilizzano software 3D ma anche più generico. Siamo convinti che con una GUI più semplice Blender si imporrebbe in termini assoluti. Del resto si può fare anche il ragionamento inverso. Gimp è molto popolare perché tutto sommato l'ambiente di lavoro è simile a quello di altri programmi commerciali , come Photoshop, a cui gli utenti sono abituati. Se l'interfaccia di Gimp fosse completamente diversa, decisamente "aliena"rispetto a quelle consuete funzionerebbe? Chissà. Forse il problema dell'open source oggi proprio quello di farsi capire da un pubblico di massa. Chi ha uno skill superiore ed è abituato a programmare o utilizzare software complessi, come gli amministratori di rete, non ha problemi a riconoscere e apprezzare la superiorità dei vari MySQL, Linux, Apache e compagnia. Ma gli altri? Forse un mondo in cui il software sarà tutto e solo open source rappresenta una meravigliosa utopia, ma pensare ad un maggiore equilibrio tra software commerciale e non deve essere invece una sorta di impegno personale che ci assumiamo anche nei confronti delle generazioni future.

NASCONDERE I CONTATTI DEL CELLULARE

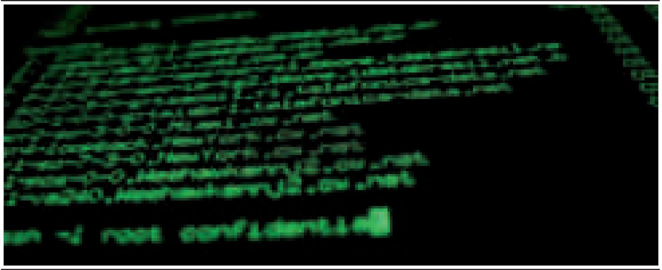
Gli smartphone sono dei veri e propri computer portatili, formato cellulare, in grado di archiviare foto, video, note e password. Ultima frontiera della tecnologia ict, ma non meno vulnerabili rispetto ad altri prodotti hitech a minacce informatiche come malware e spam, o a rischi legati a semplici perdite e furti. Per prevenire l'accesso a contenuti indesiderati e proteggere privacy e informazioni confidenziali da inconvenienti e minacce virtuali arriva da Kaspersky Lab un software innovativo: Kaspersky Mobile 9.0.



"Cresce sempre più l'utilizzo dello smartphone in Italia ed il range di età è più ampio rispetto agli anni precedenti, si parla di utenti che vanno dai 15 ai 65 - ha spiegato Alexander Moiseev, Managing Director di Kaspersky Lab Italia-. All'interno di questi dispositivi mobili vengono racchiuse tutta una serie di attività che identificano noi stessi e la nostra vita privata e lavorativa. Dalle foto alla rubrica, dalle mail agli sms. Per questo motivo -continua Alexander Moiseev- è fondamentale proteggere il nostro smartphone. I dati sullo smarrimento ed il furto sono allarmanti a livello mondiale: circa il 30% degli smartphone ogni anno, più di 250.000 negli aeroporti americani, quasi 100.000 nella metropolitana di Londra, dai 10.000 ai 15.000 nei taxi ogni mese. Ogni smartphone perduto contenente dati di lavoro, costa, in termini di esposizione a terzi e perdita dei dati, circa 2200 euro". Con la nuova funzione protezione della privacy di Kaspersky Mobile Security 9 ogni utente ha garantito il controllo esclusivo di contatti e numeri che è possibile nascondere con un semplice click e senza lasciare alcuna traccia. Chiunque venga in possesso di uno smartphone altrui sarà quindi completamente all'oscuro delle informazioni che vi sono nascoste.

Oltre all'innovativa tecnologia per la difesa della privacy Kaspersky Mobile Security 9 include anche un miglioramento delle funzioni antifurto permettendo all'utente di bloccare lo Smartphone da remoto nel caso il telefono sia stato smarrito o rubato. E' infatti sufficiente inviare un sms predefinito al proprio cellulare per bloccarlo, cancellando la memoria e permettendone via gps la localizzazione fino a ricevere una notifica con il nuovo numero della scheda sim che e' stata sostituita. Una volta bloccato in remoto, lo Smartphone può far apparire un messaggio che permettera' a chi l'ha trovato di renderlo al legittimo proprietario.

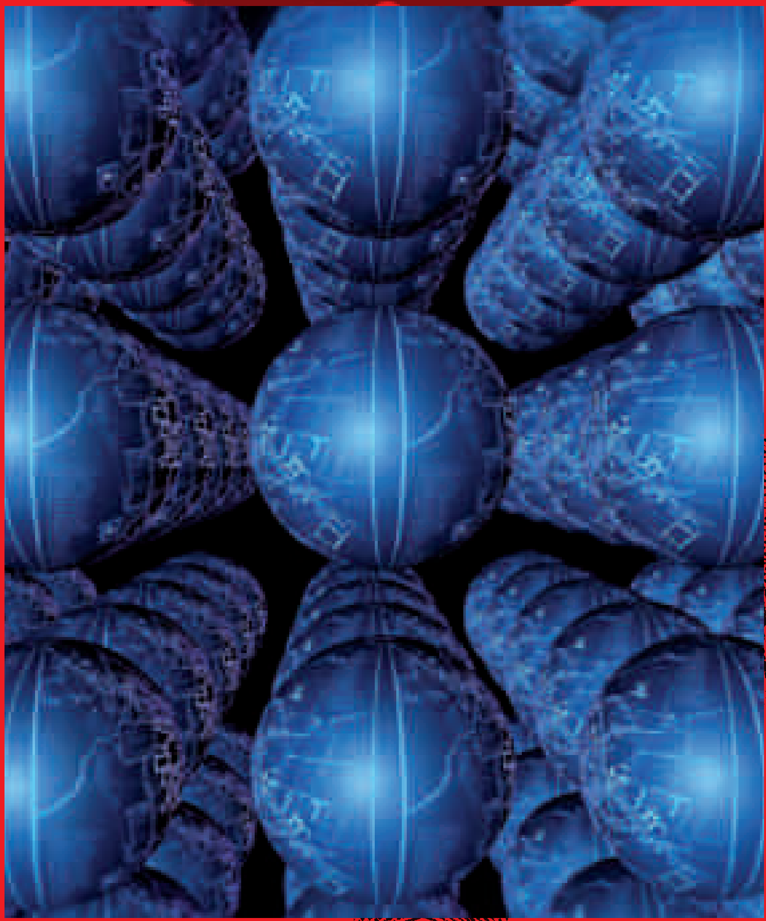
BOTNET MARIPOSA: INTRAPRENDENTI MA POCO PROFESSIONALI



Avevamo trattato nei numeri scorsi la notizia dello smantellamento della rete botnet Mariposa responsabile di avere violato conti bancari, carte di credito, username e password di una rete mondiale di 12.7 milioni di computer , appartenenti a utenti privati, aziende, agenzie governative e università di oltre 190 paesi. La rete di bot è stata disattivata il 23 dicembre 2009 grazie all'impegno congiunto di diversi esperti, agenzie e corpi di sicurezza, tra i quali Defence Intelligence, Panda Security, l'FBI e la Guardia Civil spagnola.
Il principale botmaster, conosciuto come "Netkairo" e "hamlet1917", insieme ai suoi due collaboratori, "Ostiator" e "Johnyloleante" sono stati arrestati.
"Le prime analisi indicano che i botmaster non possedevano conoscenze avanzate di hacking. Questo risulta molto preoccupante in quanto dimostra quanto sia diventato efficace e sofisticato il software di diffusione del malware, che consente a criminali senza esperienza di causare danni e perdite molto significativi", sottolinea Pedro Bustamante, Senior Research Advisor di Panda Security. Tra i principali dati emersi finora si evidenziano i seguenti elementi:
- Una volta colpiti dal client bot di Mariposa, il botmaster installava differenti malware (keylogger avanzati, banker trojan come Zeus, o trojan per l'accesso remoto, etc.) per poter realizzare azioni aggiuntive sui PC zombie.
- Il botmaster guadagnava denaro con la vendita di parti della rete di bot, installando toolbar e fornendo dati bancari per servizi online e carte di credito rubati per realizzare transazioni attraverso "muli" all'estero.
- La rete di bot Mariposa si è diffusa in modo molto efficace attraverso le reti peer-to-peer, le penne USB e le messaggerie istantanee.

Tremate, tremate, Zeus è tornato

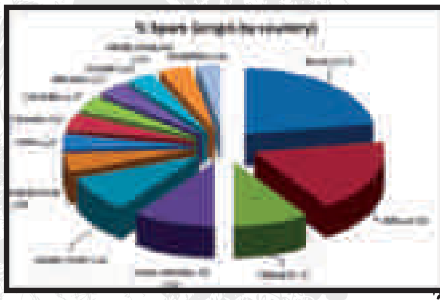
La società di sicurezza Trusteer ha affermato che il virus Zeus è tornato più potente che mai e che il numero dei computer infettati sta aumentando rapidamente. Zeus è un virus in grado di rubare dati bancari online insidiando gli utenti di computer infetti.
Il pericoloso virus infetta gli ignari utenti che usano i browser Firefox o Internet Explorer per esplorare la rete.
Il malware ruba le informazioni di login perlopiù quando chi naviga si trova su siti bancari o altri istituti finanziari. I dati sottratti vengono poi inviati a un server remoto per essere rivenduti o per essere utilizzati in modo sconveniente.
Lo scorso marzo il virus era stato per così dire sconfitto, quando alcune parti erano state messe fuori uso ed il provider che veniva utilizzato per gestire il malware era stato chiuso. Tuttavia per gli addetti ai lavori non è stato difficile ricrearlo da un'altra parte.
Gli utenti del computer devono sempre vigilare e soprattutto aggiornare regolarmente i software antivirus adottati.

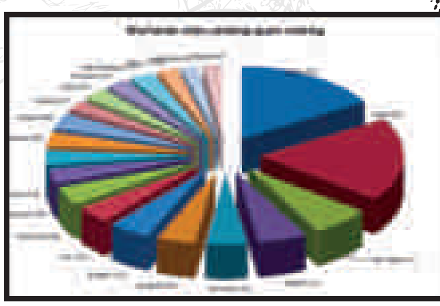


NeWS

Le classifiche dello Spam

Un'analisi di Panda Security evidenzia che Brasile, India, Corea, Vietnam e Stati Uniti sono ai vertici della classifica dei paesi fonte di spam nei primi due mesi del 2010. 5 milioni di email, analizzate dai laboratori di Panda, sono state inviate da circa un milione di indirizzi IP diversi, con una media di 5 messaggi spediti da ogni indirizzo (ogni IP potrebbe essere associato a uno o più computer). Il Brasile è al primo posto della classifica con il 13.76% di spam inviato. Messaggi di questo tipo sono utilizzati principalmente per diffondere minacce o vendere prodotti illeciti e il metodo utilizzato quale tecnica per trarre in inganno gli utenti è la promessa di video o immagini di ragazze brasiliane. Seguono l'India, con il 10.98% e la Repubblica Coreana con il 6.32%. Per quanto riguarda le città dalle quali sono state inviate email spam, Seoul è risultata la più attiva, seguita da Hanoi, New Delhi, Bogota, San Paolo e Mumbai. L'elenco completo è disponibile al link www.flickr.com/photos/panda_

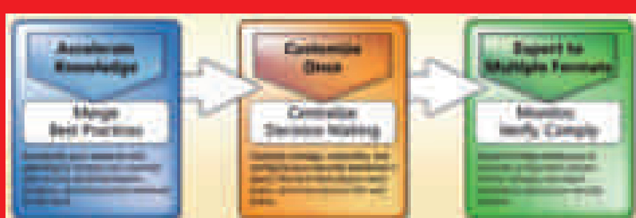




Rilasciato il tool Microsoft Security Compliance Manager

È stata rilasciata la versione definitiva del tool Microsoft che vi permette di gestire al meglio la creazione e la manutenzione centralizzata delle Security Baseline, le checklist di configurazione delle opzioni di sicurezza dei sistemi operativi:

Windows 7 Security Just Got Easier (http://technet.microsoft.com/it-it/library/cc677002(en-us).aspx)



Tra i diversi benefici si segnalano le numerose modalità di esportazione di queste baseline per automatizzare la loro distribuzione e applicazione in azienda: fogli Excel (XLS), Group Policy objects (GPOs), Desired Configuration Management (DCM) packs, Security Content Automation Protocol (SCAP).

Guido Camerlingo AKA Guiz
redazione@hackerjournal.it

DOWNLOAD A PROVA DI BOMBA

INTERNET

IN UNA GIUNGLA DI LINK CIRCOLANTI NEL WEB È IMPORTANTE TENERE SOTTO CONTROLLO IL DOWNLOAD DI FILE DAL PROPRIO SITO PER NON DEPAUPERARE PREZIOSE RISORSE DI BANDA...

Tempi duri per i nostri siti sommersi dallo spam, da tentativi di Sql Injection e anche da siti che linkano i nostri file – vere sanguisughe che non ci offrono neanche un link o un riconoscimento in cambio del download offerto.
Molti webmaster si appoggiano su server che ospitano il loro sito. Molte aziende d'hosting hanno un limite mensile di trasferimento, al termine del quale , il sito ospitato viene sospeso in attesa che passi il mese o che venga sganciata una sommetta per aumentare la banda mensile o per darci qualche altro GB di banda solo per il mese corrente.
Direi che la soluzione migliore è prendere davvero il controllo dei download, per capire se si necessita di più banda o se c'è qualcuno che ce ne succhia…
La soluzione da adottare è impedire l'accesso diretto al file HYPERLINK "http://www.miosito.it/file.zip" http://www.miosito.it/file.zip e di creare una pagina che possa "sparare" il file all'utente senza mostrargli la path e che inoltre controlli se l'utenza provenga da un altro sito.
Possiamo utilizzare facilmente la funzione header di php per mascherare la path del file. Questa funzione ci permette di nascondere la path del file ma non di impedire i download arbitrari.
Per tutelarci dai download arbitrari utilizziamo una chiave random da registrare in sessione nella pagina del download.
Vediamo lo schema:

La pagina in PhP ha lo scopo di registrare la sessione (magari registrandoci il REFERER o una chiave a nostra scelta) , verificarla e dopo aver accettato la sessione di inviare il file all'utente tramite la funzione header.
L'utilizzo del REFERER, o di una chiave a scelta, è una valida alternativa al fastidioso CAPTCHA per il download.
Atto Pratico
Passiamo all'opera creando un piccolo script in php con un Db MySQL.
Per prima cosa creiamo la struttura del database:

`N_REC_ID`	progressivo
`C_DWNNAM`	nome del download
`M_DWNDES`	descrizione download
`C_FILPHT`	path del file per il download ('download/prova.pdf')
`C_FILTYP`	la tipologia del file (Esempio 'application/pdf')

In questo tutorial ometteremo il caricamento nel db per motivi di spazio. E' sufficiente un form che alimenti il database, volendosi proprio applicare, anche un caricamento tramite ftp direttamente dall'interfaccia web.
Le pagine in php che utilizzeremo saranno due:
`downloads.php ,downloads_sel.php, downloads_act.php e conf.php`
`conf.php`
Questo sarà il nostro file di configurazione con la connessione MySQL

```
<?php
//Parametri DB
$server="localhost";
$user_n="root";
$password="pasword";
$db_name="NomeDB";
$db_type="mysql";
    //Parametri Download
    $wKeySec = 'KiaVESecre-
Ta';            //Chiave
segreta
    //Connessione Database
   $connessione = mysql_
connect($server, $user_n,
$password)
   or die("Connessione non
```

```
riuscita: " . mysql_error());
   mysql_select_db($db_name, $connessione)
   or die ("Errore nella selezione del database.");
 ?>
```

Downloads.php

In questo file listiamo tutti i downloads presenti nel database.

```
<?php
  //Includo il file di configurazione per la connessione
  include 'conf.php';
    //query di selezione nel db
  $query = "SELECT * FROM downloads";
  $result = mysql_query($query, $connessione);
  while($row = mysql_fetch_array($result)){
        echo $row['C_DWNNAM']." - <a href=\"downloads_sel.php?id=".$row['N_REC_ID']."\">Scarica</a>";
  }
  //Chiudo la connessione
  mysql_close();
?>
```

Downloads_Sel.php

In questa pagina apriamo una sessione e registriamo una variabile di sessione con una chiave criptata in MD5. Inoltre elencheremo delle informazioni sul download.

La chiave sarà composta da N_REC_ID del download + La nostra Chiave Segreta. Ovviamente è possibile modificare a piacere questa chiave. Magari aggiungendo una data, in modo da renderla ulteriormente più forte.

La nostra chiave sarà quindi: MD5(CHIAVE SEGRETA + N_REC_ID)

In questa pagina, inoltre, è presente anche il link 'Scarica' il quale fa riferimento all'ultima pagina, quella che si occupa del downloads.

```
<?php
  //Apro la session
 session_start();
 //Includo il file di configurazione per la connessione
 //e per la chiave segreta
include 'conf.php';
 //Recupero l'ID dal GET
$mId = mysql_real_escape_string($_GET['id']);
 //query di selezione nel db
 $query = "SELECT * FROM downloads where N_REC_ID = ".$mId;
$result = mysql_query($query, $connessione);
$row = mysql_fetch_array($result);
 //Registro in sessione la chiave segreta più l' id del record
 $_SESSION['key_str'] = md5($wKeySec.$row['N_REC_ID']);
 echo "<center>".$row['C_DWNNAM']."<br><a href=\"downloads_act.php?id=".$row['N_REC_ID']."\">Scarica</a></center>";
 //Chiudo la connessione
 mysql_close();
?>
```

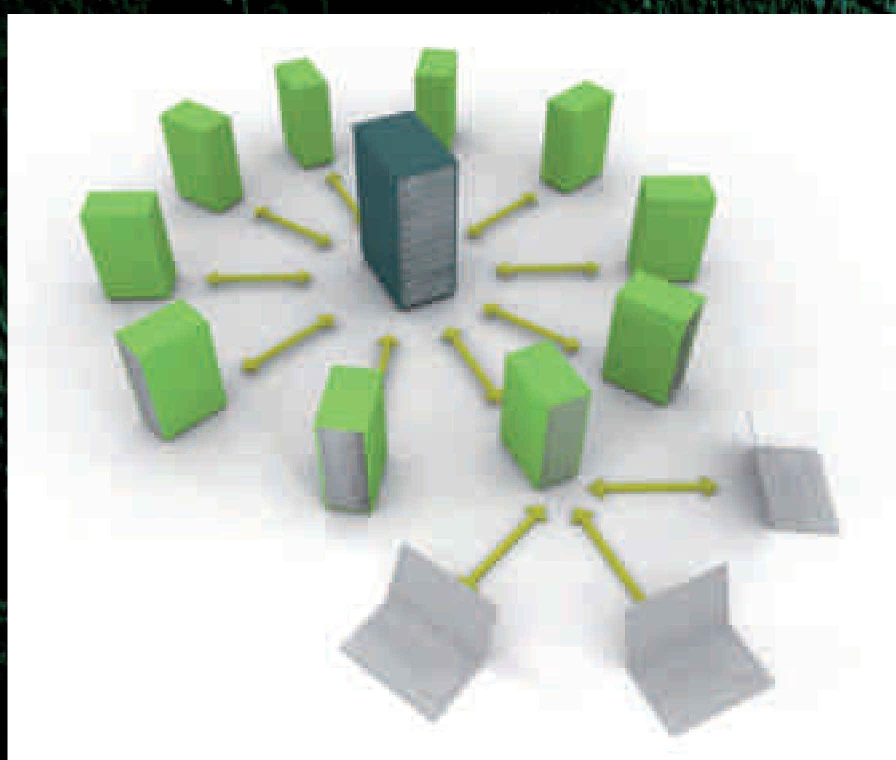
Downloads_Act.php

In questa pagina controlleremo che la chiave registrata in sessione corrisponda a quella che rigeneriamo. Se coincidono andremo a leggere il file dalla path nel db. Nel caso non coincidono la procedura terminerà. E' molto importante inserire il Content-type esatto nel database. Questo sarà indispensabile farlo in fase di caricamento del file. Per esempio se caricato tramite un form per trovare il content type sarà sufficiente

```
$FilTyp = $_FILES['NOME']['type'];
```

Questa restituirà il content type corretto (esempio per un file pdf 'application/pdf').

```
<?php
  //Apro la sessione
  session_start();
  //Includo il file di configurazione per la connessione
  //e per la chiave segreta
  include 'conf.php';
  //Recupero l'ID dal GET
  $mId = mysql_real_escape_string($_GET['id']);
  //query di selezione nel db
  $query = "SELECT * FROM downloads where N_REC_ID = ".$mId;
  $result = mysql_query($query, $connessione);
  $row = mysql_fetch_array($result);
```



```
   //CONTROLLO se l'md5 in
sessione corrisponde con il
mio calcolato
   if($_SESSION['key_str'] <>
md5($wKeySec.$row['N_REC_
ID'])){
      //Se non coincidono esco
dalla procedura
      mysql_close();
      exit;
   }
   $name = $row["C_FILPHT"];
   $type = $row["C_FILTYP"];
   header("Content-type:
$type");
   header("Content-Di-
sposition: attachment;
filename=$name");
   readfile($row['C_FILPHT']);
   //Chiudo la connessione
   mysql_close();
?>
```

Aumentiamo la protezione

Potremo in ogni caso aumentare la protezione controllando in downloads_act.php il REFERER di provenienza.

Quindi andremo a controllare se il visitatore proviene dal nostro sito o è un visitatore esterno.

in seguito andremo a creare una variabile in conf di nome wAppHostSite e la valorizzeremo con il nostro nome dominio. Per esempio 'guiz.it'

La funzione getHost (che potremo copiare nel file conf.php) estrae dal REFERER il dominio (la possiamo inserire in conf).Copiamo la funzione di controllo nel downloads_act.php. Nel caso il dominio estratto nel REFERER non coincida con quello settato nella variabile wAppHostSite termineremo lo script nella pagina.

Funzione getHost

```
function getHost($pAddr) {
    $parseUrl = parse_
url(trim($pAddr));
    return
trim($parseUrl[host] ?
$parseUrl[host] : ar-
ray_shift(explode('/',
$parseUrl[path], 2)));
}
```

Controllo

```
<?php
      //Controllo REFERER
      if(!(getHost($_
SERVER["HTTP_REFERER"]) ==
$wAppHostSite)){
            exit;
      }
?>
```



Record di prova

Di seguito un record di prova da inserire nel database.

Il dump completo del database è presente nel sorgente scaricabile al sito http://www.guido8975.it/dwn.rar

INSERT INTO `downloads` VALUES (1, 'Mio Downloads', 'Descrizione', 'download/buono.pdf', 'application/pdf');

Conclusione

L'esempio completo è disponibile sul sito http://www.guiz.it/ .

```
--
-- Struttura della tabella
`downloads`
--
CREATE TABLE `downloads` (
  `N_REC_ID` int(11) NOT
NULL auto_increment,
   `C_DWNNAM` text NOT NULL,
   `M_DWNDES` blob NOT NULL,
   `C_FILPHT` varchar(50) NOT
NULL default '',
   `C_FILTYP` varchar(50) NOT
NULL default '',
   PRIMARY KEY  (`N_REC_ID`)
) ENGINE=InnoDB DEFAULT
CHARSET=latin1 AUTO_
INCREMENT=1 ;
```

RECORD DI PROVA

Di seguito un record di prova da inserire nel database. Il dump completo del database è presente nel sorgente scaricabile al sito http://www.guido8975.it/dwn.rar

```
INSERT INTO `downloads`
VALUES (1, 'Mio Downloads',
'Descrizione', 'download/
buono.pdf', 'application/
pdf');
```



di Massimiliano Rinaldi
redazione@hackerjournal.it

P2P: UNA NORMATIVA "NEBULOSA"



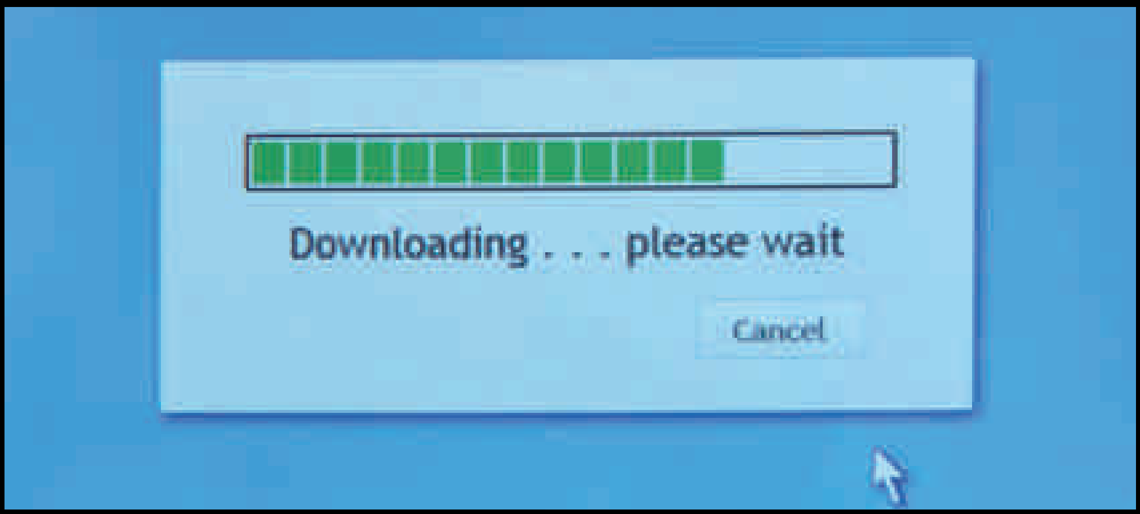
LEGGE

La recente sentenza di un tribunale spagnolo ha fatto scalpore. Ma com'è attualmente la situazione in Italia per quanto concerne il P2P?

L'incipit di questo articolo ce lo dà la notizia, pubblicata come news sul numero 200 di HJ in cui si parlava del dispositivo della sentenza emessa dal tribunale di Barcellona secondo cui i siti peer to peer non violano il diritto d'autore.
Nella sede del tribunale spagnolo si è in realtà ripresa un norma giurisprudenziale che anche in altri paesi è stata adottata in sede di giudizio: offrire link che rimandano ad altri contenuti - anche coperti da copyright, come succede nei P2P - non è illegale. La responsabilità ricade non sui collettori di link, come peraltro potrebbe essere anche Google, ma sui siti a cui rimandano quei link. Ovvero i siti che ospitano e distribuiscono, senza averne diritto, materiale protetto da copyright.
E' una sentenza che ha sollevato un grande clamore, ma che riguarda solo una delle molteplici sfaccettature del P2P la cui complessità normativa, specie in Italia, dà adito a numerosi dubbi.

LE ORIGINI DELLA LEGGE ITALIANA

C'era una volta, è proprio il caso di dirlo, la legge del 22 Aprile 1941 n.633 sulla "Protezione del diritto d'autore e di altri diritti connessi al suo esercizio". Questa legge è rimasta sostanzialmente in vigore fino al 2004 quando ci si è dovuti arrendere all'evidenza che il mondo, anche quello della proprietà intellettuale, stava cambiando in modo definitivo.
Si stava aprendo il mercato dell'ADSL ad uso privato, con la possibilità di scaricare film in formato divx (mpeg4) e musica in formato MP3. Una rivoluzione.
E' proprio in questo contesto storico, sotto il governo Berlusconi, che sotto la supervisione dell'allora Ministro per i Beni e le Attività Culturali Giuliano Urbani viene tratteggiato un testo di legge destinato ad innovare la materia.
Il testo di quello che sarebbe diventato il primo Decreto Legge 72/04 fu motivo di grande scontro tra maggioranza e opposizione, in



L'articolo 171-ter prevevede la reclusione da sei mesi a tre anni e la multa da euro 2.582 a euro 15.493.

particolare da parte dei Verdi col loro capogruppo Fiorello Cortina.
Nella sua prima stesura il testo del DL 72/04 teneva valido il principio, per determinare i casi di punibilità, della "finalità di lucro" che ispira, di fatto, la legge del 22 aprile 1941 n. 633.
Probabilmente si era scelta questa linea per non discostarsi troppo dalla legge cardine e semplificare le cose, soprattutto in considerazione che si stava trattando una materia ai più misteriosa, internet e lo scambio dei file, quindi in questi casi il buon senso consiglia di modificare il meno possibile.
Molto probabilmente nell'intento di chi ha a suo tempo disegnato questo Decreto c'era solo la banale volontà di punire, con gli stessi criteri della legge n.633, altri contesti di violazione del diritto d'autore venuti alla luce in seguito all'evoluzione tecnologica.
Dunque nulla di complicato, solo un'operazione di maquillage.
Tuttavia in sede di presentazione l'accezione "per finalità di lucro", dopo molte polemiche, venne modificata in "per trarne profitto" cosicché la versione presentata alle Camere dell'articolo 171-ter recitava così:

"E' punito, se il fatto è commesso per uso non personale, con la reclusione da sei mesi a tre anni e con la multa da cinque a trenta milioni di lire (all'epoca c'erano ancora le lire) chiunque per trarne profitto…".
Se prima l'articolo 171-ter parlava di punibilità di azioni commesse "a scopo di lucro" la sostituzione del "Lucro" con il "Profitto" finì per equiparare, di fatto, il testo dell'articolo 171 ter comma 1 della legge 41/63 all'articolo 171-bis, ingenerando ulteriore confusione.

Il Decreto venne approvato in fretta e furia. In realtà il testo non convinceva proprio nessuno, né maggioranza, né opposizione, ma fu proprio il Ministro Urbani a chiedere il "sacrificio" del voto immediato di una legge necessaria impegnandosi ad introdurre dopo le opportune modifiche.

IL MOSTRO

Il Decreto Legge così approvato il 22 Marzo 2004, N. 72, ha rappresentato un motivo di dubbi e controversi sul fronte della sua applicazione pratica perché contradditorio e di difficile lettura anche per i giudici. Insomma uno strumento inadeguato che ha complicato molto le cose.
Per cercare di ristabilire un certo ordine con legge del 31 marzo 2005, n. 43 sono state introdotte una serie di modifiche la più importante delle quali è la sostituzione della formula "a fini di lucro" con "per trarne profitto" nel comma 1 dell'articolo 171 –ter, in una sorta di ritorno all'antico…
Nel box principale pubblichiamo il testo integrale proprio dell'articolo 171-ter che riguarda da vicino proprio la regolamentazione e la punibilità del P2P.

OK, MA COSA RISCHIO?

Il testo di legge attuale non ha risolto tutte le contraddizioni della legge, comunque la giurisprudenza è servita a fare un po' di chiarezza in quello che, nel tempo, è diventato un testo aggiustato e rattoppato che forse andrebbe smantellato completamente e riscritto.
La sostituzione della parola profitto con lucro ha messo un po' più al sicuro gli utenti.
Infatti dovrebbe essere al sicuro chi scarica file a titolo personale.
Però non è altrettanto al sicuro chi li condivide, infatti l'articolo Art. 171 recita:

Salvo quanto disposto dall'art. 171-bis e dall'articolo 171-ter è punito con la multa da euro 51 a euro 2.065 chiunque, senza averne diritto, a qualsiasi scopo e in qualsiasi forma:
a) riproduce, trascrive, recita in

pubblico, diffonde, vende o mette in vendita o pone altrimenti in commercio un'opera altrui o ne rivela il contenuto prima che sia reso pubblico, o introduce e mette in circolazione nello Stato esemplari prodotti all'estero contrariamente alla legge italiana; a-bis) mette a disposizione del pubblico, immettendola in un sistema di reti telematiche, mediante connessioni di qualsiasi genere, un'opera dell'ingegno protetta, o parte di essa...
Dunque per il combinato disposto dei due articoli chi scarica per uso proprio non è punibile ma chi condivide file protetti sicuramente sì, anche se non con pene detentive ma solo con sanzioni pecuniarie. Però il paradosso è evidente. Se nessuno condivide finisce il peer to peer.
Quindi questa normativa non tiene forse conto dell'aspetto più interessante del file sharing che è quello della condivisione del sapere, non necessariamente legato a contenuti protetti da copyright. Spaventare l'utente con norme poco chiare può solo servire a rallentare un processo comunque auspicabile e molto interessante per la crescita comune.

Cambia notevolmente il discorso nel caso si realizzi una condivisione di contenuti proptetti per finalità di lucro.
E qui non si intende solo la vendita di contenuti dietro compenso, oppure il download di contenuti poi masterizzati, duplicati e distribuiti, ma anche la presenza, su un sito di materiale protetto che viene "regalato" agli utenti ma che procura un vantaggio al titolare del sito attraverso i ricavi pubblicitari che il maggiore traffico del sito produce.
Insomma la finalità del guadagno diventa centrale.

ART. 171-TER

1. È punito, se il fatto è commesso per uso non personale, con la reclusione da sei mesi a tre anni e con la multa da euro 2.582 a euro 15.493 chiunque a fini di lucro: a) abusivamente duplica, riproduce, trasmette o diffonde in pubblico con qualsiasi procedimento, in tutto o in parte, un'opera dell'ingegno destinata al circuito televisivo, cinematografico, della vendita o del noleggio, dischi, nastri o supporti analoghi ovvero ogni altro supporto contenente fonogrammi o videogrammi di opere musicali, cinematografiche o audiovisive assimilate o sequenze di immagini in movimento; b) abusivamente riproduce, trasmette o diffonde in pubblico, con qualsiasi procedimento, opere o parti di opere letterarie, drammatiche, scientifiche o didattiche, musicali o drammatico-musicali, ovvero multimediali, anche se inserite in opere collettive o composite o banche dati;
c) pur non avendo concorso alla duplicazione o riproduzione, introduce nel territorio dello Stato, detiene per la vendita o la distribuzione, o distribuisce, pone in commercio, concede in noleggio o comunque cede a qualsiasi titolo, proietta in pubblico, trasmette a mezzo della televisione con qualsiasi procedimento, trasmette a mezzo della radio, fa ascoltare in pubblico le duplicazioni o riproduzioni abusive di cui alle lettere a) e b);
d) detiene per la vendita o la distribuzione, pone in commercio, vende, noleggia, cede a qualsiasi titolo, proietta in pubblico, trasmette a mezzo della radio o della televisione con qualsiasi procedimento, videocassette, musicassette, qualsiasi supporto contenente fonogrammi o videogrammi di opere musicali, cinematografiche o audiovisive o sequenze di immagini in movimento, od altro supporto per il quale è prescritta, ai sensi della presente legge, l'apposizione di contrassegno da parte della Società italiana degli autori ed editori (S.I.A.E.), privi del contrassegno medesimo o dotati di contrassegno contraffatto o alterato;
e) in assenza di accordo con il legittimo distributore, ritrasmette o diffonde con qualsiasi mezzo un servizio criptato ricevuto per mezzo di apparati o parti di apparati atti alla decodificazione di trasmissioni ad accesso condizionato;
f) introduce nel territorio dello Stato, detiene per la vendita o la distribuzione, distribuisce, vende, concede in noleggio, cede a qualsiasi titolo, promuove commercialmente, installa dispositivi o elementi di decodificazione speciale che consentono l'accesso ad un servizio criptato senza il pagamento del canone dovuto.
f-bis) fabbrica, importa, distribuisce, vende, noleggia, cede a qualsiasi titolo, pubblicizza per la vendita o il noleggio, o detiene per scopi commerciali, attrezzature, prodotti o componenti ovvero

I CD

La stessa legge all'articolo 171-bis si occupa di regolamentare l'attività di duplicazione di DVD e CD contente materiale protetto e recita:

1. Chiunque abusivamente duplica, per trarne profitto, programmi per elaboratore o ai medesimi fini importa, distribuisce, vende, detiene a scopo commerciale o imprenditoriale o concede in locazione programmi contenuti in supporti non contrassegnati dalla Società italiana degli autori ed editori (SIAE), è soggetto alla pena della reclusione da sei mesi a tre anni e della multa da euro 2.582 a euro 15.493. La stessa pena si applica se il fatto concerne qualsiasi mezzo inteso unicamente a consentire o facilitare la rimozione arbitraria o l'elusione funzionale di dispositivi applicati a protezione di un programma per elaboratori. La pena non è inferiore nel minimo a due anni di reclusione e la multa a euro 15.493 se il fatto è di rilevante gravità.

Quindi sembrerebbe lecito effettuare copie private di opere protette pur non possedendone l'originale (quindi masterizzare il film scaricati ad uso personale su DVD) però la locuzione "per tarne profitto" è piuttosto "sibillina". Se io duplico un film o un videogioco (o lo masterizzo dopo averlo scaricato da internet) per evitare di acquistare l'originale ne traggo un profitto? Per certo versi sì, perché risparmio una somma di denaro. Sarebbe stato molto meglio e tranquillizzante se il legislatore si fosse espresso con la locuzione "a fini di lucro" come accade per l'articolo 171-ter.

presta servizi che abbiano la prevalente finalità o l'uso commerciale di eludere efficaci misure tecnologiche di cui all'art. 102-quater ovvero siano principalmente progettati, prodotti, adattati o realizzati con la finalità di rendere possibile o facilitare l'elusione di predette misure. Fra le misure tecnologiche sono comprese quelle applicate, o che residuano, a seguito della rimozione delle misure medesime conseguentemente a iniziativa volontaria dei titolari dei diritti o ad accordi tra questi ultimi e i beneficiari di eccezioni, ovvero a seguito di esecuzione di provvedimenti dell'autorità amministrativa o giurisdizionale;

h) abusivamente rimuove o altera le informazioni elettroniche di cui all'articolo 102 quinquies, ovvero distribuisce, importa a fini di distribuzione, diffonde per radio o per televisione, comunica o mette a disposizione del pubblico opere o altri materiali protetti dai quali siano state rimosse o alterate le informazioni elettroniche stesse.

2. È punito con la reclusione da uno a quattro anni e con la multa da euro 2.582 a euro 15.493 chiunque:

a) riproduce, duplica, trasmette o diffonde abusivamente, vende o pone altrimenti in commercio, cede a qualsiasi titolo o importa abusivamente oltre cinquanta copie o esemplari di opere tutelate dal diritto d'autore e da diritti connessi;

a-bis) in violazione dell'art. 16, a fini di lucro, comunica al pubblico immettendola in un sistema di reti telematiche, mediante connessioni di qualsiasi genere, un'opera dell'ingegno protetta dal diritto d'autore, o parte di essa;

b) esercitando in forma imprenditoriale attività di riproduzione, distribuzione, vendita o commercializzazione, importazione di opere tutelate dal diritto d'autore e da diritti connessi, si rende colpevole dei fatti previsti dal comma 1;

c) promuove o organizza le attività illecite di cui al comma 1.

3. La pena è diminuita se il fatto è di particolare tenuità.

4. La condanna per uno dei reati previsti nel comma 1 comporta:

a) l'applicazione delle pene accessorie di cui agli articoli 30 e 32-bis del codice penale;

b) la pubblicazione della sentenza in uno o più quotidiani, di cui almeno uno a diffusione nazionale, e in uno o più periodici specializzati;

c) la sospensione per un periodo di un anno della concessione o autorizzazione di diffusione radiotelevisiva per l'esercizio dell'attività produttiva o commerciale.

5. Gli importi derivanti dall'applicazione delle sanzioni pecuniarie previste dai precedenti commi sono versati all'Ente nazionale di previdenza ed assistenza per i pittori e scultori, musicisti, scrittori ed autori drammatici.

MOBILE/MEDIO

di Ing. Mauro Meneghin
redazione@hackerjournal.it

GPS TRACKING CON IL CELLULARE

TRACCIAMENTO

Raccogliere le informazioni di un percorso via GPS per poi visualizzarlo su PC? Basta usare GPSLogger e un po' di programmazione.

A volte mi piace fare qualche escursione in montagna, avventurandomi per sentieri che non conosco. Talvolta, tornato a casa, ho cercato di capire che percorso avevo fatto consultando una cartina, con 100 dubbi su dove e a che ora ero passato esattamente. Ora esistono i navigatori satellitari per fortuna, però non tutti offrono la possibilità di tenere traccia del percorso effettuato e di avere il dettaglio sugli istanti esatti di percorrenza.

Che ne dite invece di usare un cellulare per raccogliere i dati di posizione e successivamente visualizzare il percorso effettuato direttamente sul pc?

REQUISITI

-un cellulare android con GPS e una memoria uSD
-un pc con lettore SD e connessione a internet
-un adattatore da uSD a SD (meno di 5 euro)

Il software da installare sul cellulare è GPSLogger disponibile gratuitamente http://www.androlib.com/android.application.com-mendhak-gpslogger-ipxp.aspx oppure http://apps.doubletwist.com/GPS-Logger-for-Android/-2727505723369031487

Una volta lanciato andate su Menu->Settings (vedi Fig. 1) e selezionate "Log to GPX" (in questo modo i log verranno salvati in files con estensione .gpx all'interno della memoria uSD del cellulare), selezionate inoltre un valore per "Time before logging", in modo che i dati di posizione vengano salvati con la frequenza desiderata, io uso 60 secondi e 0 metri come "Distance before logging".

A questo punto qualche minuto prima di uscire di casa cliccate su "Start logging", noterete che il logging non parte subito quindi abbiate pazienza, l'applicazione sta provando a individuare la posizione di partenza cercando un segnale su parecchi satelliti e più ne troverà maggiore sarà la precisione. Dovreste iniziare a vedere le coordinate della vostra posizione come in Fig. 2..
Una volta tornati a casa togliete la scheda uSD dal cellulare e inseritela in un adattatore uSD->SD (vedi Fig. 3), inserite poi l'adattatore nello slot per memorie SD del computer. Questa memoria utilizza un file system vfat e dovrebbe essere automaticamente riconosciuta (su opensuse viene montata automaticamente in /media/disk). Notere un file del tipo /mount_dir/GPSLogger/20100413.gpx in cui il nome del file corrisponde alla data annomesegiorno.gpx in cui è stato effettuato il logging.
Si tratta di un file xml con questo formato:

```
<?xml version="1.0"?>
<gpx … >
  <time>2010-04-
13T16:22:05+02:00</time>
  <bounds />
  <trk>
    <trkseg>
```

```
        <trkpt lat="44.35" 
lon="12.15">
          <ele>121.0</ele>
          <src>gps</src>
          <sat>4</sat>
          <time>2010-04-
13T16:22:05+02:00</time>

        </trkpt>
...
```

in cui ogni punto (chiamato anche track point o abbreviando trkpt) viene salvato con due attributi lat=latitudine, lon=longitudine, mentre altre informazioni vengono fornite come elementi figli (tempo, altitudine, ...).

Per visualizzare facilmente questi file ho creato un script in php a cui potete accedere sul mio server: http://g1ld0.is-a-geek.net/gps_track/ Basta fare l'upload del file, lo script provvede a fare il parsing del xml, ne estrapola ora, longitudine, latitudine e se presenti anche altitudine e velocità e costruisce il vostro tracciato come notate in Fig. 4.

Mi sono accorto navigando tra i sorgenti di GPSLogger che l'autore prevede di realizzare una nuova versione del programma con la possibilità di visualizzare la mappa del tracciato direttamente sul cellulare, tuttavia per ora tale possibilità non è ancora attiva, inoltre dal codice sembra che voglia richiedere l'autentificazione degli utenti con password per usare questo servizio.

UN PO' DI PROGRAMMAZIONE

DALVIK

Gran parte dei programmi scritti per Android sono scritti in Java, in realtà non si trovano tutte le classi standard del Java, ma si trovano delle classi aggiuntive che consentono di sviluppare agevolmente applicazioni che utilizzano tutte le funzionalità hardware dei cellulari (es. fotocamera, bluetooth, telefonia, SMS, GPS, touch sceen, ...). La virtual machine che interpreta questi programmi si chiama Dalvik.
Se qualcuno fosse interessato allo sviluppo esiste un plugin per eclipse che facilita notevolmente lo sviluppo di applicazioni per Android e include anche un emulatore per eseguire le applicazioni sul pc senza la necessità di avere collegato un cellulare.

Vediamo molto sinteticamente come fa GPSLogger ad utilizzare la funzionalità GPS.
Viene utilizzata la classe android.location.LocationManager che serve per richiedere aggiornamenti sulla posizione corrente.
Gli aggiornamenti sulla posizione vengono inviati a una classe GeneralLocationListener che implemanta l'interfaccia android.location.LocationListener. Grosso modo il codice che richiede gli aggiornamenti è questo:

```
android.location.LocationManager
gpsLocationManager;
gpsLocationManager
= (LocationManager)
getSystemService(Context.
LOCATION_SERVICE);
gpsLocationManager.requestLoc
ationUpdates(LocationManager.
GPS_PROVIDER,minimumSeconds
* 1000,minimumDistance,
gpsLocationListener);
gpsLocationManager.addGpsStatus
Listener(gpsLocationListener);
...
```

durante il logging verrà richiamata in modo automatico la funzione (int event) dell'oggetto gpsLocationListener con i nuovi dati sulla posizione corrente.
Per chi fosse interessato rimando

DEFINIZIONE DEI TRACK POINT, <GPX><TRK><TRKSEG><TRKPT>

ATTRIBUTO	ELEMENTO FIGLIO	SIGNIFICATO	OBBLIGATORIO
lat		latitudine	si
lon		longitudine	si
	ele	altitudine [m]	no
	course	direzione	no
	speed	velocità [m/s]	no
	src	dati via gps o wifi	si
	sat	numero di satelliti	no
	time	istante di rilevazione	si

di Massimiliano Rinaldi
redazione@hackerjournal.it

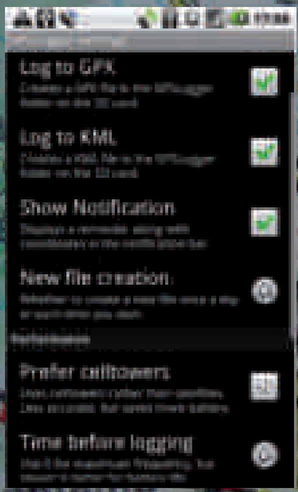


Figura 1 - Menu->Settings.

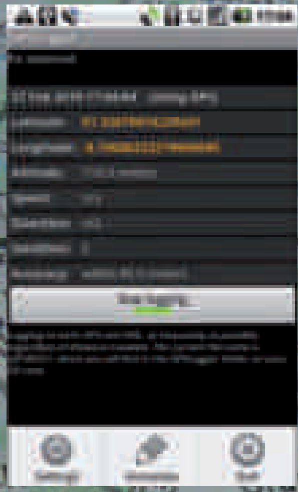


Figura 2 - Le coordinate.



Figura 3 - L'dattatore uSD->SD

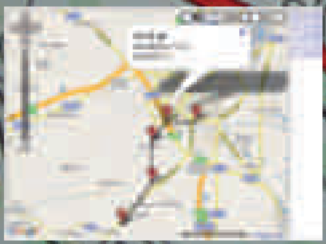


Figura 4 - Il tracciato.

alla documentazione ufficiale di android.location.LocationManager al sito: http://developer.android.com/reference/android/location/LocationManager.htm

Script PHP

Nello script PHP per il parsing del file xml sul server web, una volta letto il contenuto del file nella stringa $data, ho costruito un oggetto xml con la funzione simplexml_load_string() e poi leggo tutti i nodi <trkpt>:

```
  // costruisce un oggetto xml
$xml = simplexml_load_string($data);
  // cerca tutti i nodi <gpx><trk><trkseg><trkpt>
foreach ($xml->trk->trkseg->trkpt as $node)
{
  $lat=$node['lat'];   // latitudine
  $lon=$node['lon']; //longitudine
…
```

JavaScript e Google Maps

Per rappresentare il tracciato con le Google Maps, per ogni nodo trovato ho fatto in modo che nel codice JavaScript prodotto uscisse (semplifico un po' per brevità):

```
// crea il punto con latitudine lat e longitudine lon
point = new GLatLng(lat,lon);
// crea il marker del punto
marker = new GMarker(point);
map.addOverlay(marker);
path.push(point);
```

inoltre cliccando sui track points faccio aprire una finestra con informazioni aggiuntive come l'altitudine e la velocità.
Infine per tracciare il percorso con una poligonale le api di Google Maps mettono a disposizione la funzione GPolyline(), che prende come argomento l'array di punti path riempito prima:

```
var polyline=new GPolyline(path,'#000000',2,1.0);
map.addOverlay(polyline);
```

Lorenzo `lolunix` Finestrino
redazione@hackerjournal.it

Algoritmo RSA

SICUREZZA

L'RSA è uno degli algoritmi di cifratura asimmetrica più noti. La sua sicurezza è data dalla sua difficoltà di fattorizzazione di numeri molto grandi. Scopriamo come funziona!

Innanzitutto si scelgono due numeri primi come ad esempio P e Q e si calcola il loro prodotto **N:N = P*Q.**
Occorre quindi calcolare quanti numeri sono compresi tra 1 e N-1 che sono relativamente primi con N (due numeri sono relativamente primi o primi tra loro se il loro massimo comune divisore è 1).

Questa funzione è nota come **toziente di Eulero** ed è solitamente indicata con la lettere greca minuscola Φ (phi).
Per esempio Φ(9) = 6, poiché 1,2,4,5,7 e 8 sono relativamente primi con 9.

Quindi se N è primo, Φ(N) sarà N – 1 e se N è il prodotto di esattamente due numeri primi P e Q, allora **Φ(P*Q) = (P – 1) * (Q – 1). Tutto questo risulta utile in quanto l'algoritmo RSA richiede il calcolo di Φ(N).**

Occorre scegliere a caso una chiave di cifratura che soddisfi la seguente equazione, dove S è un intero qualsiasi: E * D = S * Φ(N)+1. Questa equazione può essere risolta con l'algoritmo di Euclide esteso.

L'algoritmo di Euclide è un antico algoritmo che consente di calcolare molto rapidamente il massimo comune divisore (MCD) di due numeri. Si prende il maggiore dei due numeri e lo si divide per quello minore, tenendo conto soltanto del resto. Poi si divide il numero minore per il resto e si ripete il processo fintanto che il resto non è pari a zero.

L'ultimo valore del resto prima di arrivare a zero è il massimo comune divisore dei due numeri di partenza. Questo algoritmo è piuttosto veloce, con un tempo di esecuzione **O(log10N)**. Ciò significa che il numero di passaggi richiesti dall'algoritmo dovrebbe essere uguale al numero di cifre del numero maggiore.

Prendiamo come esempio il calcolo del MCD di **7253** e **120** riportato nella seguente tabella.

Questa è suddivisa in tre colonne: A, B, R: nella prima troviamo il primo numero (7253), nella seconda il numero minore (120), nell'ultima il resto. Com'è facile capire, ad ogni passaggio il vecchio B diventa il nuovo A e il vecchio R diventa il nuovo B.
Viene calcolato R finche è il resto è zero, l'ultimo valore di R prima della zero è il massimo comune divisore.

A	B	R
7253	120	53
120	53	14
53	14	11
14	11	3
11	3	2
3	2	1
2	1	0

In questo caso il massimo comune divisore è 1, questo significa che 7243 e 120 sono relativamente primi tra loro.

Lorenzo `lolunix` Finestrino
redazione@hackerjournal.it



Ci torna utile pertanto **l'algoritmo di Euclide esteso** che permette di trovare due interi J e K: **J * A + K * B = R** quando il massimo comune divisore tra A e B è proprio R. Questo si ottiene quando si utilizza l'algoritmo di **Euclide a ritroso**, in questo caso contano i quozienti, di seguito riporto i calcoli per l'esempio precedente:

7253 = 60 * 120 + 53
120 = 2 * 53 + 14
53 = 3 * 14 + 11
14 = 1 * 11 + 3
11 = 3 * 3 + 2
3 = 1* 2 + 1

Aiutandoci con l'algebra si possono spostare i termini in modo che il resto si trovi a sinistra prima dell'uguale quindi avremo:

53 = 7253 – 60 * 120
14 = 120 – 2 * 53
11 = 53 – 3 * 14
3 = 14 -1 * 11
2= 11 -3 * 3
1 = 3-1 * 2

È chiaro iniziando dal basso che

1 = 3 -1 * 2
Nella riga sopra questa tuttavia c'è **2 = 11 – 3 * 3** che dà una sostituzione per 2:

1 = 3 – 1 * (11 -3 * 3)
1 = 4 * 3 -1 * 11

Nella riga ancora superiore **3 = 14 -1 * 11** si sostituirà con 3:

1 = 4 * (14 – 1 * 11) – 1 * 11
1 = 4 * 14 – 5 * 11

Nella riga ancora più in alto **11 = 53 – 3 * 14** richiede una ulteriore sostituzione:

1 = 4 * 14 – 5 * (53 – 3 * 14)
1 = 19 * 14 – 4 * 53

Saliamo ancora di una riga **14 = 120 -2 * 53** applicando un'altra sostituzione:

1 = 19 * (120 -2 * 53) -5 * 53
1 = 19 * 120 -43 * 53

E infine la riga in cima con **53 = 7253 – 60 * 120** che richiede un'ultima sostituzione:

1 = 19 * 120 – 43 * (7253 – 60 – 120)
1 = 2599 * 120 – 43 * 7253
2599 * 120 + -43 * 7253 = 1

Ciò mostra che J e K sarebbero rispettivamente 2599 e – 43.

Supponendo che i valori di P e Q siano 11 e 13, N sarebbe 143. Perciò **Φ(N) = 120 = (11 -1) * (13 -1)**. Questo perchè7253 è relativamente primo con 120 e costituisce pertanto un eccellente valore per E.

Tutto questo magari ci ha fatto perdere di vista il nostro obbiettivo che era quello di trovare un valore per D che soddisfacesse l'equazione: **E * D = S * Φ(N) + 1**. Semplificando:

D * E + S * Φ(N) = 1
D * 7253 +/- S * 120 + 1

Usando i valori ottenuti dall'algoritmo di Euclide esteso, risulta evidente che D = - 43. Il valore per S in realtà non conta e questo significa che i calcoli matematici fatti sono modulo Φ(N) o modulo 120.
Questo a sua volta significa che un

valore equivalente positivo per D è 77, perchè 120 – 43 = 77.
Il valore quindi può essere inserito nell'equazione di prima:

$$E * D = S * \Phi(N) + 1$$
$$7253 * 77 = 4654 * 120 + 1$$

I valori per N ed E sono distribuiti come chiave pubblica, mentre D è mantenuto segreto come chiave privata.
P e Q sono scartati. Le funzioni di cifratura e decifratura sono piuttosto semplici:

Cifratura: $C = M \wedge E \ (modN)$
Decifratura: $M = C \wedge D(modN)$

per esempio se il messaggio M è 98 la cifratura sarebbe $98 \wedge 7253 = 76(mod143)$.

Il testo cifrato sarebbe 76 ed appare evidente immaginare che solo chi conosce il valore di D può in questo caso decifrare il messaggio e riottenere il numero 98 a partire dal numero 76, semplicemente con: $76 \wedge 77 = 98(mod143)$.

Ovviamente se il messaggio M è più grande di N deve essere suddiviso in porzioni che siano più piccole di N.
Questo processo è reso possibile dal **teorema di Eulero** il quale afferma che se M e N sono primi tra loro e M è il minore, quando questo è moltiplicato per sé stesso $\Phi(n)$ volte e diviso per N il resto sarà sempre 1:

Se mcd(M,N) = 1 e M < N allora $M \wedge \Phi(N) = 1 \ (modN)$. Poiché tutto è calcolato in modulo N vale anche quanto segue, per il modo in cui funziona la moltiplicazione nell'aritmetica modulo:

$$M \wedge \Phi(N) * M \wedge \Phi(N) = 1 * 1(modN)$$
$$M \wedge 2 \Phi(N) = 1 \ (modN)$$

Questo processo potrebbe essere ripetuto S volte per ottenere:

$$M \wedge S \Phi(N) = (modN)$$

Se si moltiplicano entrambi i membri per M il risultato è:

$$M \wedge S \Phi(N) * ; = 1 * M(modN)$$
$$M \wedge S \Phi(n)+1 = M(modN)$$

Questa equazione è il cuore dell'algoritmo RSA. Un numero M elevato a una potenza modulo N produce ancora il numero M di partenza, si tratta in sostanza di una funzione che restituisce il proprio stesso input il che non ha grande interesse in sé. Ma se questa equazione potesse essere suddivisa in due parti separate allora una parte potrebbe essere usata per cifrare e l'altra per decifrare , producendo ancora il messaggio originale.

Questo si può fare trovando E e D che moltiplicati tra loro diano come risultato S per $\Phi(N) + 1$. Il valore ottenuto lo si può sostituire nella precedente equazione:

$$E * D = S * \Phi(N) +1$$
$$ME * D = M \ (modN)$$

che equivale a:

$$M \wedge E \wedge D = M(modN)$$

che può essere suddivisa in due parti:

$$ME = C \ (modN)$$
$$CD = M(modN)$$

La sicurezza dell'algoritmo RSA dipende dal fatto che D sia mantenuto segreto ma poiché N ed E sono entrambi valori pubblici, se N può essere fattorizzato nei P e Q originali allora $\Phi(N)$ può essere facilmente calcolato con (P – 1) * (Q -1) e poi D può essere determinato con l'algoritmo di Euclide esteso.

Per mantenere la sicurezza computazionale, le dimensioni delle chiavi RSA devono essere scelte con il migliore algoritmo di fattorizzazione. Attualmente il migliore è NFS (Number Field Sieve).
Questo algoritmo ha un tempo subesponenziale buono ma per determinare RSA a 2.048 bit non è ancora molto performante.



di Raffaele Rambaldi
Raffaele_Rambaldi@hotmail.com

Prevenire lo sniffing

RETI

Lo sniffer è uno strumento molto potente che può essere utilizzato per ascoltare e vedere ciò che, in realtà, si vorrebbe proteggere. Per capire come evitare l'attacco da parte di uno sniffer dobbiamo imparare a conoscere i meccanismi che regolano il traffico in rete.

Abbiamo imparato ad usare uno sniffer. Lo abbiamo installato e siamo in grado di raccogliere le informazioni che ci interessano tra quelle in transito sulla rete.
Adesso, almeno per una volta, dismettiamo i panni dell'hacker e posizioniamoci sull'altra sponda, dove siedono i famosi esperti di sicurezza.
La priorità per un esperto di sicurezza è proteggere i dati, sempre e comunque. Purtroppo non si può sempre far ricorso al tunnelling vpn, crittografia, chiavi esotiche di cifratura e quant'altro. In qualche caso ciò che viene trasmesso deve passare in chiaro e l'eventuale uso di uno sniffer, da parte di un non avente diritto, deve essere assolutamente evitato.
Per capire come proteggere il traffico di rete da occhi indiscreti dobbiamo partire dall'inizio, ossia da come è fatta una rete.

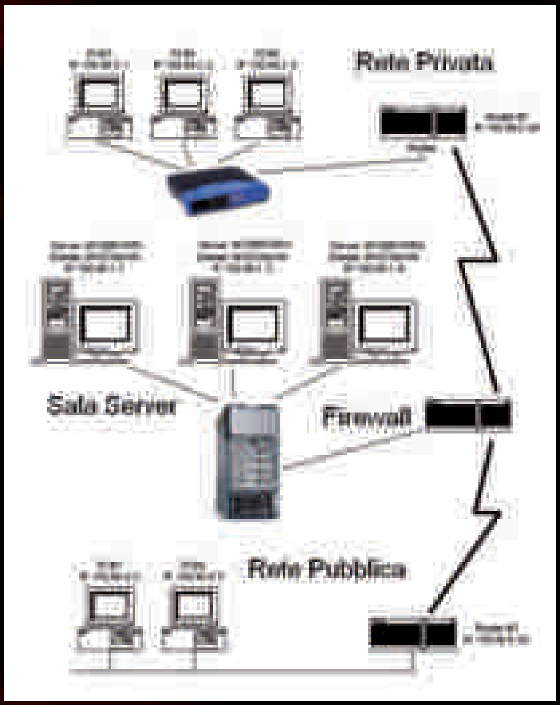
SEGMENTI DI RETE

Un segmento di rete consiste in un gruppo di macchine che condividono traffico a bassissimo livello, che tipicamente coincide con il segmento di collisione. Una volta le reti erano formate da un unico cavo di rete coassiale che aveva un inizio ed una fine racchiusi tra due "tappi", ma erano frequenti i problemi perché bastava che un cavo fosse "toccato" dalla signora delle pulizie per interromperne il funzionamento.
Successivamente le reti si sono evolute e grazie all'introduzione del multiport repeater i problemi, soprattutto dovuti ai collegamenti, sono diminuiti ma in entrambi i casi il funzionamento è il medesimo.
Tutte le schede di rete parlano ed ascoltano, trasmettendo e ricevendo dati sulla medesima frequenza dello stesso filo, anche se in tempi diversi, dell'ordine di millisecondi. Quando due schede di rete trasmettono contemporaneamente si genera una collisione ed entrambe ritrasmettono i propri dati ripetendo successivamente il medesimo segnale. Tanto è minore il numero di collisioni in percentuale rispetto al numero di

pacchetti trasmessi, tanto maggiore è la performance della rete.
Va da se che quando i nodi sono troppi avviene ciò che sappiamo in un luogo troppo affollato, ossia si crea un rumore di fondo che rende praticamente impossibile la conversazione tra due persone. Infatti, superato il numero di 30 schede di rete che insistono sul medesimo segmento di collisione, le performance degradano in maniera esponenziale al crescere del traffico.
Per questo motivo, sono stati inventati gli switch, detti anche bridge intelligenti, che anziché ripetere i segnali a basso livello verso tutti i nodi ed attendere che le collisioni regolino il traffico, si interpongono come un filtro lasciando passare solo i dati che interessano il nodo che si trova su quella determinata porta, creando di fatto un circuito virtuale tra chi trasmette e chi riceve.
Gli switch di questo tipo gestiscono al proprio interno una tabella ARP e sanno esattamente su quale porta si trova quella determinata scheda. Se un nodo vuole aprire una sessione, i segnali vengono copiati solo sulla porta interessata, nascondendo di fatto il traffico agli altri nodi.

IL PROTOCOLLO TCP/IP

Esiste un altro metodo per nascondere i dati. Attraverso l'uso del TCP/IP un segmento di rete può coincidere con una subnet di indirizzi gestita da un router. Definire i confini di una subnet vuol dire rendere appartenenti al medesimo segmento un gruppo di host non solo a livello fisico ma al livello di protocollo.
Quando si parla di infrastruttura di rete occorre sempre avere in mente il modello ISO/OSI. Bisogna posizionarsi mentalmente al livello di riferimento, considerando il livello fisico sul gradino più basso ed il livello applicativo al livello più alto, passando per diversi strati gestiti da uno o più protocolli di comunicazione.
Detto questo, fate molta attenzione perché due o più subnet gestiti al livello 3 del protocollo TCP/IP possono tranquillamente condividere lo stesso segmento fisico di rete gestito dal secondo o primo livello.
Infatti, è possibile utilizzare il medesimo switch per gestire due diverse subnet, senza fare ricorso ad opzioni esotiche di qualche switch più costoso come ad esempio le LAN virtuali. In questo caso, i computer che appartengono ai due diversi gruppi potranno vedersi tra di loro ma non trasversalmente con i computer dell'altro gruppo. Per consentire l'apertura di un canale di comunicazione tra due computer appartenenti a due subnet diverse occorre configurare un router che generi un'appropriata tabella di instradamento e che abbia la possibilità di raggiungere un indirizzo TCP/IP di entrambe le subnet.
Non occorre che il router abbia due schede di rete e che ciascuna di esse abbia l'indirizzo TCP/IP definito all'interno di ciascuna delle due subnet. E' possibile anche configurare sulla medesima scheda del router due indirizzi TCP/IP ciascuno appartenente ad una delle due subnet ed il gioco è fatto.



La rete può diventare molto complessa. In quella esemplificata in questa figura, qualcuno sulla rete pubblica potrebbe anche posizionare uno sniffer ma sarebbe pressoché impossibile visualizzare il traffico che va dalla sala server alla rete privata.

BARRIERE HARDWARE

Per rendere sicuro un segmento bisogna considerare prima gli altri segmenti a cui il primo è collegato. In un ufficio dove ci sono determinate regole per accedere, come ad esempio una sala server, è difficile installare uno sniffer senza essere scoperti. Diverso è invece il caso di una sala pubblica dove è consentito il libero accesso. Pensiamo tipicamente ad una Università, dove esistono segmenti di Facoltà che dovrebbero rimanere privati e segmenti pubblici, come ad esempio la biblioteca, dove entra e si collega chiunque.
Ovviamente, l'uso di strumenti come Firewall e simili sono praticamente un obbligo. Seguendo i fattori di costo e di performance sarebbe molto più semplice collegare tutte le macchine tra loro attraverso apparati economici, ma dal punto di vista della sicurezza questa soluzione è un pugno allo stomaco.
In questo caso occorre racchiudere tutte le macchine ad accesso pubblico in uno spazio ben delimitato, creando un solo ed unico collegamento verso la sala server, adeguatamente presidiato da un firewall.
In questo caso, i dati che viaggiano sul segmento sicuro dovrebbero quasi certamente restare inviolati anche se ci sono moltissimi fattori in gioco, come ad esempio la tipologia degli edifici, la distanza tra gli apparati di rete ed il numero di computer utilizzati. In ogni caso, ho usato il condizionale perché nel mondo dell'informatica, quando si parla di sicurezza, il condizionale è obbligatorio.

:: POSTA ::

ROUTER D-LINK DVA-G3670B: IL ROUTER DEL MISTERO

Grazie per l'attenzione, vi scrivo perché disperato :-(
Premessa, sembra che questo router sia di proprietà Infostrada e Wind e che alla D-LINK ITALIA, non abbiano autorizzazioni per l'assistenza perché le Aziende sopra citate hanno bloccato il prodotto. Ad oggi il ruoter D-Link DVA-G3670B sembra non esistere presso la sede Italiana, mentre si trova liberamente in commercio su vari siti di e-commerce. Chiamando l'assistenza D-Link al numero 199 400 057 il call-center mi ha risposto che non esiste…
Ad oggi, indagando sui vari siti, a anche su sito della D-LINK mondo, non sono riuscito a trovare il prodotto, qualcuno può darmi una dritta ?
Grazie Mille per L'attenzione.
Luca – Bologna

Non siamo in grado di darti una risposta, giriamo la tua accorata richiesta di aiuto ai lettori, magari qualcuno si è già trovato nella medesima situazione.

LIBERO PENSIERO/1

Leggo l'articolo di Stallman e il primo pensiero che mi attraversa la mente, ancora immerso nella lettura, è: come fa a girare il mondo, vivere, mangiare... se nulla fa per scopo di lucro, e allora la mente vola d'istinto all'immagine dei santoni che incantano, ammaliano la gente con bei discorsi e fanno la loro fortuna con le benevole offerte dei seguaci. Di certo non è opportuno il paragone, potrebbe essere persino offensivo, data la certo differente dimensione etica, ma lo spunto ha un suo significato... Di certo Stallman è una grande mente, ma io non mi sono mai fidato di chi fa la propria fortuna puntando l'indice su altri e sparare a zero su Microsoft e Bill Gates è facile, di certo gratuito, ma non così giusto. Il mercato è libero, la pubblicità fa vivere migliaia di persone ed è pubblicità pure ogni conferenza di Stallman. Parliamo finalmente di noi, dei nostri pregi. Cambiamo davvero il mondo in qualcosa di migliore, di veramente pacifico, non esprimendo più ironici dissensi, o, peggio, rancori o odio per opinioni o strategie differenti. Accettiamo tutti i pensieri pure quelli di Bill Gates, della politica di Microsoft e dimostriamo che le cose giuste sono convivenza, comprensione, umiltà. Sì, perché se davvero siamo convinti che ci siano alternative migliori a programmi a pagamento bastano le persone che ne parlano, la rete che diffonde le opinioni, per avere successo. So per esperienza diretta che ogni buon appassionato di computer ha molte persone che gli chiedono consiglio ed hanno fiducia della sua opinione e questo basta a dare fiducia e preferenza a determinati programmi, piuttosto che altri. Possibile che non si possa fare una battaglia davvero in libertà, senza dovere accanirsi o infangare qualcun altro, o qualcos'altro? Insomma, trovo di cattivo gusto i continui riferimenti nel free e nell'open a Bill Gates. Non è Dio, non un angelo, ma nemmeno un demone e se è arrivato dove è arrivato, partendo da un garage, beh, delle buone idee deve averle avute e in questo mondo c'è ancora la possibilità di fare qualcosa di buono con delle buone idee.

Enrico P.

Sottoscriviamo quello che dici: libero pensiero, libera iniziativa e tanta tolleranza e apertura mentale. Peccato che spesso il messaggio non venga recepito. Per quanto riguarda Stallman è evidente che col tempo il suo credo è diventato quasi uno show commerciale che porta in giro per il mondo con grande sagacia e dedizione. Però è altrettanto vero che senza Stallman probabilmente non ci sarebbe stato Linux, e senza Linux il mondo informatico sarebbe oggi più chiuso e arroccato sulle sue posizioni, forse meno progredito. Quindi tutti noi dobbiamo qualcosa a Stallman. Se poi nel tempo da sviluppatore geniale si è convertito in profeta da palcoscenico tutto sommato glielo possiamo concedere. Contrariamente a chi parla senza avere mai fatto nulla, Stallman porta con sé un grande patrimonio di idee, innovazione e libertà. E scusate se è poco…

PARADOSSI DEL MONDO OPEN

Fa strano leggere che il Mondo dotato del più straordinario mezzo di comunicazione di tutti i tempi (passati) sia un Mondo sempre più chiuso. Fa strano anche perché sono le stesse statistiche sulla diffusione dei sistemi operativi che contraddicono questa affermazione: il sistema operativo più diffuso sui server di rete non è proprietario. Potrebbe essere l'eccezione che conferma la regola, forse, ma è un fatto da considerare. LAMP (o la sua contaminazione WAMP) risultano essere uno standard ormai consolidato. Ma allora, sono giustificate tutti questi allarmi di intrusione nella privacy, di costrizione all'acquisto, di pensiero unico dominante? La risposta è come sempre in uno dei fondamenti dell'economia, dal valore assimilabile al più noto "qui prodest": "follow the money". Richard Stallman è di fatto uno show man, al pari di molti altri show man conosce bene il repertorio ed il canovaccio. Il repertorio può cambiare, potrebbe essere etico, religioso o politico, ma tutti gli show man hanno una cosa in comune: convertono in denaro dei concetti condivisi eliminando le sfumature. La loro scusante è sempre la stessa: è una questione (etica/religiosa/politica) di principio. Vi diranno che il Mondo è bianco o nero, e di seguire questa semplice regola. Che la mettano in pratica o no poco importa. Ne trarranno comunque un vantaggio economico.
Il Mercato invece segue principi diversi, molteplici e variegati, segue pensieri dominanti e laterali nel contempo, consuetudini e stravaganze, mode e revivals, logici e paradossali perché il Mercato rifugge da una

:: POSTA ::

standardizzazione, dalla catalogazione, dall'etichettatura. Il Mercato è uno, nessuno e centomila. Il Mercato accoglie il software libero ed open source e allo stesso tempo il software proprietario. Li vedremo lottare ad armi pari o a colpi bassi (GPL docet) ancora negli anni a venire. Facile profezia? Vedremo! Non dimentichiamo che il vero signore di tutti gli dei, per gli antichi Greci, era (è e sarà) il Tempo.

Giorgio Drei

Caro Giorgio, (iniziamo questa risposta come se fosse la posta del cuore di Confidenze perché in effetti la tua mail ci è parsa davvero passionale, scritta col cuore). La tua disamina crediamo sia stata solleticata dall'editoriale apparso sul numero 199 "Logiche e meccanismi di consumo".
Quello che dici a proposito dei meccanismi di mercato è assolutamente sottoscrivibile, oltre che esplicato in modo davvero efficace. E' proprio il mercato che determina vinti e vincitori in modo equanime. Nello specifico la nostra analisi era infatti rivolta proprio al questi nuovi sistemi di diffusione dei contenuti su dispositivi cellulari che il mercato sembra premiare. Va da sé che, come dicevi in apertura, l'utilizzo di software aperto lato server è sempre più diffuso.
Per Stallman invece di rimandiamo alla risposta data alla mail precedente. Anche qui c'è del vero, ma per molti della redazione Stallman prima che un profeta è un uomo di grande intuizione e grandi idee. Lo abbiamo incontrato più volte e abbiamo imparato ad apprezzarne pregi e difetti. Gli vogliamo bene come se fosse uno di famiglia, quindi non potremmo mai parlare male di lui, ma rispettiamo il tuo pensiero.

H COME HACKER

Che spesso si faccia un uso sbagliato delle parole non è una novità. E finché questo accade davanti ad un caffè o dal salumiere, e le cavolate che si dicono vengono ascoltate da un pubblico di nicchia (che magari le dimenticherà appena finita la conversazione) può anche passare. Ma se la confusione è fatta da giornali, televisioni o ancora peggio su internet, tutti strumenti che si rivolgono ad un pubblico di notevole mole e non sempre in grado di distinguere il corretto dalla panzana, allora viene da storcere il naso, ed anche parecchio. Mi è capitato qualche giorno fa di ascoltare un servizio di un telegiornale nazionale (quindi con un grande volume di ascolto) nel quale si parlava di phishing, furto di dati e pericolo su internet. E fin qui tutto bene, l'iniziativa è lodevole, se pensiamo ad esempio alla casalinga che guarda il Tg e può aprire gli occhi su quali sono i pericoli cui potrebbe andare incontro il figlio quando naviga in rete. Il problema è stato l'uso sbagliato della parola hacker fatto all'interno di quello stesso servizio. Hacker, una parola bella, che parla di persone belle che con l'umiltà dei bistrattati cercano di dare il proprio contributo a migliorare le cose.
Per usare questa parola in maniera corretta sarebbe sufficiente anche solo tradurla letteralmente dall'inglese ("colui che taglia, che smonta"). E invece no, si usa hacker in sostituzione di criminale o ladro o truffatore, magari perché una parola inglese luccica di più. Ma il luccichio non deve servire a distruggere una filosofia, un'etica, un mondo di valori ed un modo di pensare che nulla hanno a che fare con il crimine. Ancora peggio, potrebbe esserci qualcuno che crede che hacker sia proprio il termine tecnico da utilizzare per identificare i criminali informatici. Ripenso alla casalinga di prima, alla prossima volta che sentirà la parola hacker, a cosa potrebbe succedere se vedesse suo figlio leggere questa stessa rivista.
Non pretendo puntate di talk show dedicate all'hacking e all'etica hacker (che non sarebbero male, anzi), ma già adoperare le parole in maniera corretta, o non adoperarle affatto se non si conosce il loro significato, sarebbe un passo avanti. Chiudo trascrivendo la definizione della parola Hacker trovata in cima alla pagina dell'editoriale di un vecchio numero di HJ:

Persona che si diverte ad esplorare i dettagli dei sistemi di programmazione e come espandere le loro capacità, a differenza di molti utenti, che preferiscono imparare solamente il minimo necessario.

Dionigi

Quello che scrivi è vero. Del resto alle lene hanno fatto anche di peggio con un servizio che mostrava "in esclusiva" un pericoloso hacker che non è passato inosservato nel nostro forum. Lì forse c'era più "dolo" ovvero quel servizio è stato confezionato ben sapendo che non era rappresentativo del mondo hacker ma solo per fare un po' di scalpore su un target poco esperto. Viceversa nelle redazioni dei telegiornali c'è quasi sempre buona fede ma l'approssimazione deriva dal fatto che purtroppo i giornalisti che ci lavorano non hanno una preparazione così specifica, così verticale, per tutti gli argomenti. Scrivere e parlare in modo corretto, "scientifico", di argomenti molto tecnici non è semplice. Si rischia di cadere nel banale e quando confezioni un prodotto generalista come il telegiornale spesso non ci sono né i mezzi, né il tempo per approfondire tutte le notizie.
Ne parliamo a ragion veduta perché molti di noi arrivano da esperienze diverse, anche in piccole Tv e l'approccio è esattamente questo. Diverso il discorso quando si va sulla stampa specializzata. Lì tutto dovrebbe essere assolutamente irreprensibile. Ma le vie del sapere sono tante e tali da non escludere nessuno "scivolone"...
In coda vogliamo riportare l'aneddoto di un nostro redattore che scrivendo qualche tempo fa un articolo su una rivista di Yacht (già proprio Yacht) ha scoperto a sue spese che la generica vernice impermeabilizzante dello scafo descritta nell'articolo era in realtà "resina epossidica" e così avrebbe dovuto chiamarsi. Inutile dire che i lettori si sono molto adirati...

Giovanni Federico - giovanni.federico@isek.it
Fabio `BlackLight` Manganiello - blacklight86@gmail.com

PARTE II

CORSO DI PROGRAMMAZIONE IN C

LINGUAGGI ESAURITA LA NECESSARIA INTRODUZIONE TEORICA AFFRONTATA NELLA PRIMA PARTE DEL CORSO, ILLUSTREREMO IN QUESTE PAGINE ALCUNI DEI CONCETTI PIÙ IMPORTANTI DEL LINGUAGGIO CHE, INSIEME A QUANTO VISTO FINORA, PERMETTERANNO DI REALIZZARE IL NOSTRO PRIMO APPLICATIVO IN C.

ALGORITMI

Obiettivo principale di qualsiasi percorso elaborativo è definire con precisione una successione di eventi volta alla risoluzione di determinati algoritmi. Lo stesso termine algoritmo è intrinsecamente legato ai medesimi concetti di successione sequenziale o ricorsiva di azioni.

DEFINIZIONE 7

Definiamo algoritmo un procedimento descritto da una sequenza di azioni atte alla risoluzione di un problema assegnato e caratterizzato da inscindibili presupposti di finitezza, descrivibilità, comprensibilità e riproducibilità. La risoluzione di un algoritmo da parte del Calcolatore si traduce nell'esecuzione di una successione (sequenziale e/o ricorsiva) di azioni, ciascuna delle quali con un inizio ed una fine ben determinata e calcolabile.

In merito alla complessità ed alla calcolabilità degli algoritmi risulta, inoltre, necessario offrire la seguente:

DEFINIZIONE 8

Per studiare la trattabilità e la calcolabilità di algoritmi esistono due tipi di complessità computazionale a cui far riferimento: spaziale (quantità di memoria richiesta per immagazzinare i dati necessari) e temporale (tempo richiesto per l'esecuzione dell'algoritmo e la finalizzazione del risultato, spesso identificato anche con il termine "efficienza").

Ai tempi d'oggi la complessità spaziale non costituisce, come un tempo, un parametro di rilievo in quanto i moderni elaboratori si avvalgono, nella stragrande maggioranza dei casi, dell'utilizzo di grossi quantitativi di memoria.
Tuttavia per alcune applicazioni, come gli algoritmi usati in contesti di intelligenza artificiale, la complessità spaziale può diventare un parametro critico, in quanto in tali contesti ricorsivi la complessità spaziale può facilmente crescere con andamento esponenziale al crescere della dimensione del problema.
Ad esempio, le richieste in termini di memoria occupata da un algoritmo intelligente per il gioco degli scacchi possono aumentare di un fattore 10 se aggiungiamo solo una riga e una colonna alla nostra scacchiera. In contesti di tale complessità è facile intuire che lo spazio occupato e l'efficienza dell'algoritmo sono parametri strettamente legati, in quanto i tempi richiesti per l'allocazione e la deallocazione di tanta memoria impattano in modo sensibile sul tempo complessivo di esecuzione dell'algoritmo.
L'efficienza è invece un parametro di assoluta importanza che va considerato in riferimento al contesto nel quale il problema viene risolto. Efficienza, pertanto, non significa considerare il dato temporale in senso assoluto ma collocare quest'ultimo nell'ambiente circostante.
Un algoritmo risulta trattabile se offre risposte in tempi consoni al contesto di riferimento. Sarà invece più efficiente se, negli stessi contesti, offrirà tempi di esecuzione e finalizzazione del risultato minori rispetto ad ulteriori

algoritmi trattabili.
Rispetto a quanto appena detto diamo quindi la seguente:

DEFINIZIONE 9

Identifichiamo con f(n) il tempo di esecuzione di un processo elaborativo, dove n è l'insieme dei dati di input.

Diremo pertanto che un algoritmo gode, ad esempio, di una complessità temporale n3 se il tempo speso per arrivare al risultato dell'elaborazione è pari al cubo della dimensione dell'input (n). Chiarito questo, possiamo offrire la seguente, più generale:

DEFINIZIONE 10

Definiamo e denotiamo come O(f(n)) la complessità temporale di un algoritmo il cui limite superiore del numero di operazioni spese dallo stesso con un input "n" di dati sia f(n), da cui: $O(f(n)) = \{ g(n): \exists\ c, n_0 \mid 0 < g(n) < cf(n)\ \forall\ n > n_0 \}$.

Da qui snodiamo tre tipologie di analisi tese ad individuare quelle operazioni che rappresentino il procedimento dominante dell'intero algoritmo considerato. Avremo pertanto:
- L'analisi del caso migliore in cui la nostra preoccupazione sarà quella di identificare il tempo di esecuzione dell'algoritmo quando "n" rappresenta un insieme di dati di facile trattamento (dimensioni discrete). In termini ingegneristici, nella quasi totalità dei casi, questo tipo di analisi non fornisce nulla di significativamente valido.
-L'analisi del caso medio dove la complessità dei dati da trattare è di difficoltà intermedia.
-L'analisi del caso peggiore quando si presentano difficoltà elevate nel trattamento dei dati. Questo tipo di analisi risulta molto utile nella definizione del tempo massimo che l'algoritmo può impiegare. È importante precisare che non risulta di particolare interesse la dimensione dei dati trattati dal processo bensì come questi crescono all'aumentare degli stessi. In altre parole, particolare attenzione sarà rivolta all'ordine di grandezza di f(n) nella condizione limite. In termini matematici, quando questa tenderà ai suoi asintoti per input sufficientemente grandi. L'analisi asintotica della complessità di un algoritmo consente di restringere ulteriormente il campo ai procedimenti predominanti prima menzionati. Facciamo un esempio: Consideriamo due algoritmi finalizzati alla risoluzione di uno stesso problema con due distinte complessità temporali h(n) e j(n). Nel caso in cui risulti h(n) asintoticamente minore rispetto a j(n) esisterà una "n" (dimensione dell'input) oltre cui l'ordine di grandezza del tempo di esecuzione del primo algoritmo [h(n)] è inferiore rispetto al secondo [j(n)]. Concludiamo riportando a titolo esemplificativo una tabella contenente le dimensioni dell'input e la complessità di tipo esponenziale su un elaboratore capace di gestire un milione di istruzioni al secondo (1 MIPS):

	n = 10	n = 20	n = 30	n = 50
n	0,00001 sec.	0,00002 sec.	0,00003 sec.	0,00005 sec.
n^2	0,0001 sec	0,0004 sec.	0,0009 sec.	0,00025 sec.
n^3	0,001 sec.	0,008 sec	0,027 sec.	0,125 sec.

COSTRUTTI DI CONTROLLO

I concetti legati ad un ben determinato modo di identificare e definire successioni per la risoluzione di algoritmi in un qualsiasi linguaggio di programmazione sono raggruppati in tre distinti modi di affrontare il problema.
È infatti possibile utilizzare un blocco di istruzioni scritto come una sequenza di precise operazioni, singole strutture selettive e costrutti iterativi.
Da questo offriamo pertanto il seguente:

TEOREMA 1

DI BÖHM-JACOPINI - 1966
Algoritmi di qualsiasi complessità possono essere espressi utilizzando indistintamente una singola sequenza di istruzioni, una singola struttura selettiva oppure una singola struttura iterativa.

Definire le azioni elaborative del Calcolatore in C significa adoperare determinati statement (enunciati) che, a seconda dell'impostazione che si vuole dare alla risoluzione degli AR, possono essere dei tipi che vedremo di seguito.

COSTRUTTI SELETTIVI

Come il nome stesso suggerisce, gli enunciati selettivi consentono la definizione di azioni elaborative in base al valore di una determinata espressione booleana. In altri termini si valuta l'espressione di riferimento ed in base al risultato di quest'ultima (true o false) si sceglie se eseguire o meno una istruzione od un blocco di istruzioni.
In questa categoria rientrano gli enunciati "if-else" e "switch". Il primo è un controllo condizionale che consente di valutare l'espressione e, se vera, eseguire un determinato blocco di istruzioni, altrimenti quanto delimitato dall'else. Qualora non fosse definito quest'ultimo statement, l'unico riferimento sarà l'espressione riferita dall'if. In questo caso, se risultasse falsa, nessuna istruzione sarebbe eseguita. Vediamo un rapido esempio:

```
int a, b;
a = b = 20;
if (a == b){ fprintf(stdout, "a e b sono uguali");}
```

Nell'esempio proposto è effettuato un confronto tra due variabili di tipo intero. Nel caso in cui risultino uguali sarà stampato a video l'apposito messaggio. Volendo contemplare anche l'ipotesi in cui i valori non siano uguali, semplicemente, inseriremo un blocco else come di seguito:

```
if (a == b){ fprintf(stdout, "a e b sono uguali");}
else { fprintf(stdout, "a e b non sono uguali"); }
```

Esiste un ulteriore modo di effettuare un confronto selettivo di tipo if-else in C chiamato "if in linea" (o operatore ternario) la cui forma è (controllo) ? <espressione 1 > : <espressione 2>. Tra parentesi tonde indicheremo l'espressione che dovrà essere oggetto di analisi: se questa risulterà vera sarà restituito quanto definito in <espressione 1>, in caso contrario, <espressione 2>. L'esempio seguente chiarirà maggiormente il concetto:

```
int a, b, c;
a = 5; b = 10;
c = (a > b) ? a : b;
fprintf(stdout, "il valore più grande è %d", c);
```

È naturalmente possibile annidare più blocchi if-else:

```
int a, b;
a = 3; b = -4;
if (a > b){
	if (a > 0){ fprintf(stdout, "a è maggiore di b ed è un numero positivo"); }
	else { fprintf(stdout, "a è maggiore di b"); }
} else { fprintf(stdout, "b è' maggiore di a"); }
```

Malgrado fin qui le abbiamo utilizzate, ricordiamo che è possibile omettere le parentesi graffe nel caso in cui si debba eseguire una singola istruzione piuttosto che un blocco. Per effettuare scelte multiple è possibile concatenare più blocchi if-else al fine di contemplare più opzioni: nell'esempio di seguito una semplice implementazione:

```
int a = 1;

if (a == 1)
	fprintf(stdout, "Prima opzione");
else if (a == 2)
	fprintf(stdout, "Seconda opzione");
else if (a == 3)
	fprintf(stdout, "Terza opzione");
else
	fprintf(stdout, "Opzione non contemplata");
```

In alternativa possiamo utilizzare lo statement su scritto (switch) che ben si presta a questi utilizzi e rappresenta un ulteriore modo di eseguire vari blocchi di istruzioni in funzione del valore dell'espressione definita. L'esempio appena proposto, utilizzando lo switch, diventa:

```
int a = 3;
switch(a){
	case 1:
		fprintf(stdout, "Prima opzione");
		break;
	case 2:
		fprintf(stdout, "Seconda opzione");
		break;
	case 3:
		fprintf(stdout, "Terza opzione");
		break;
	default:
		fprintf(stdout, "Opzione non contemplata");
}
```

L'istruzione "break" ci consente di interrompere l'esecuzione dello switch una volta eseguito il singolo blocco di istruzioni definito da una delle eventualità previste (case). Incontreremo di nuovo questa keyword nel prosieguo di questa trattazione, non appena avremo necessità di uscire da un ciclo. Concludiamo pertanto definendo formalmente la sintassi del costrutto "if-else" nella sua forma classica:

DEFINIZIONE 11

STATEMENT IF-ELSE

if (espressione di controllo) {
< blocco di istruzioni nel caso in cui l'espressione di controllo risulti verificata (TRUE) >;
}
else {
< blocco di istruzioni nel caso in cui l'espressione di controllo non risulti verificata (FALSE) >;
}

COSTRUTTI ITERATIVI

Gli statement iterativi consentono di eseguire un determinato blocco di istruzioni in modo reiterato oppure fintantoché una determinata espressione risulta vera. In questo caso, stabilendo un'opportuna "condizione", la ripetizione delle istruzioni terminerà solo quando questa risulterà soddisfatta. I cicli costituiscono un'entità fondamentale della programmazione strutturata e consentono di esprimere al meglio le capacità di elaborazione della macchina. Permettendo di eseguire più volte un determinato blocco di istruzioni rendono particolarmente agevole l'implementazione su calcolatore di algoritmi anche molto complessi. È importante segnalare fin da ora che nella totalità dei casi l'utilizzo di un tipo di ciclo piuttosto che un altro non deve intendersi come impossibilità da parte dell'uno o dell'altro di soddisfare le operazioni richieste, bensì di ottimizzare e spesso rendere più leggibile l'intero sorgente. Il linguaggio C dispone di vari cicli, di seguito li analizzeremo tutti offrendo per ognuno di essi qualche esempio. Questo ci permetterà di capire meglio gli utilizzi e le modalità di esecuzione degli stessi.

IL CICLO FOR

Il primo ciclo oggetto della nostra trattazione sarà quello di tipo enumerativo o ciclico ovvero quello da preferirsi ogni qualvolta che il numero di ripetizioni dei blocchi d'operazioni da effettuare è noto a priori.
La keyword del linguaggio preposta alla definizione dello stesso è "for".
L'inizializzazione del ciclo è effettuata mediante l'utilizzo di uno statement composto da un'istruzione di assegnazione attraverso la quale viene fornito il valore iniziale alla variabile che fungerà da contatore del ciclo, una condizione rappresentante un'espressione booleana atta ad identificare la terminazione ed, infine, un'apposita istruzione che definirà come la variabile utilizzata come contatore debba cambiare il suo valore ad ogni ripetizione.
Nella quasi totalità dei casi quest'ultima istruzione rappresenta un incremento o un decremento del valore della variabile contatore. A tal fine si utilizzano gli operatori di post e pre incremento/decremento trovati come allegati alla prima parte di questo corso nell'apposita tabella disponibile sul sito della rivista in formato pdf.
In base a quanto finora scritto possiamo pertanto offrire la seguente:

DEFINIZIONE 12

IL CICLO FOR

for (< contatore >; < condizione >; < aggiornamento contatore >) {
< istruzioni >
}

Nell'immagine di seguito riportata è schematizzato il funzionamento del ciclo for.



Il funzionamento del ciclo for.

Un tipico esempio di applicazione del ciclo for è riscontrabile nel calcolo del

fattoriale di un numero naturale come notiamo nel seguente esempio:

```
int n = 10; // Il numero per il quale si intende effettuare il fattoriale
int fattoriale = 1; // Inizializzazione della var. che conterrà il fattoriale
int contatore; // Il contatore per il ciclo for

for(contatore = n; contatore > 1; contatore--){
        fattoriale *= contatore;
}

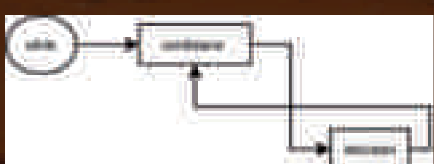
fprintf(stdout, "Il fattoriale del numero %d è %d.", n, fattoriale);
```

IL CICLO WHILE

Il ciclo while è il tipo di interazione più semplice del linguaggio ed è composto da un blocco di istruzioni che vengono eseguite ciclicamente fintanto che l'espressione di controllo risulta verificata.

Questo aspetto è di primaria importanza perché l'esecuzione del while è subordinata alla valutazione di quest'ultima espressione. L'interruzione del ciclo, inoltre, si verificherà quando la medesima assumerà il valore booleano FALSE (ovvero, non risulterà più verificata).

Da questo risulta facile immaginare che il ciclo può terminare solo se all'interno del blocco di istruzioni referenziato da esso siano presenti statement che modifichino il valore di verità dell'espressione di controllo. Qualora questo non accadesse il ciclo non potrebbe ovviamente concludersi dando luogo ad imbarazzanti errori durante l'esecuzione del nostro applicativo.



Il funzionamento del ciclo while.

Spese queste poche parole introduttive possiamo pertanto dare la seguente:

DEFINIZIONE 13

IL CICLO WHILE

```
while ( < condizione > ) {
        < istruzioni >
}
```

Come abbiamo detto poc'anzi la scelta di un ciclo piuttosto che un altro per la risoluzione di un determinato algoritmo non avviene perché l'uno può fare ciò che l'altro non può. In virtù di ciò, a titolo esemplificativo, offriamo il calcolo del fattoriale di un numero utilizzando, piuttosto che il for, il ciclo while.
Il lettore noterà autonomamente l'incremento di righe di codice necessario per esprimere lo stesso algoritmo con un ciclo diverso da quello che meglio si presta a questi scopi (in questo contesto while piuttosto che for). Da qui, di nuovo, il richiamo relativo alla scelta del ciclo migliore rispetto agli utilizzi che se ne devono fare.

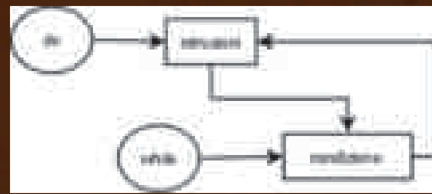
```
int n = 10;
int fattoriale = 1;
int contatore = 0;

while (contatore != n) {
        fattoriale *= (n - contatore);
        contatore++;
}

fprintf(stdout, "Il fattoriale del numero %d è %d.", n, fattoriale);
```

IL CICLO DO-WHILE

Vi sono determinate situazioni in cui è necessario eseguire un blocco di istruzioni all'interno di un ciclo indipendentemente dal valore dell'espressione di controllo. In altri termini esistono scenari all'interno dei quali è necessario eseguire almeno una volta un ciclo.
In questi casi risulta particolarmente utile avvalersi del costrutto do-while che, a differenza del while, verifica l'espressione condizionale alla fine. Il blocco istruzioni viene quindi prima eseguito e poi valutato.
A questo punto, il blocco sarà ancora eseguito finché la condizione risulterà verificata.



Il funzionamento del ciclo do-while.

Diamo pertanto la seguente:

DEFINIZIONE 14

IL CICLO DO-WHILE

```
do {
        < istruzioni >
} while ( < condizione > );
```

Per le stesse motivazioni viste nel while, offriamo ancora una volta lo stesso esempio del fattoriale di un numero naturale, stavolta con il ciclo do-while:

```
int n = 10;
int fattoriale = 1;
int contatore = 0;

do {
        fattoriale *= (n - contatore);
        contatore++;
} while (contatore != n);

fprintf(stdout, "Il fattoriale del numero %d è %d.", n, fattoriale);
```

CICLI NESTED

Come per i costruttivi selettivi è naturalmente possibile annidare cicli all'interno di altri cicli. Di seguito riportiamo un esempio con un ciclo for la cui logica tornerà utile nelle prossime trattazioni relative alla manipolazione delle matrici. Precisiamo inoltre che quanto di seguito descritto è applicabile con qualunque ciclo supportato dal linguaggio.

```
int i, j, k;
k = 1;

for(i = 0; i < 10; i++){
    for(j = 0; j < 10; j++) {

        if(k < 10)
            fprintf(stdout, "00%d ", k);
        else if (k < 100)
```

```
            fprintf(stdout, "0%d ", k);
        else
            fprintf(stdout, "%d ", k);
      k++;
      }
      fprintf(stdout, "\n");
}
```

Risultato di questa elaborazione sarà la stampa a video dei primi 100 numeri ordinati in una tabella 10 x 10 (vedi immagine).



Tabella 10 x 10 creata con due for annidati.

LOOP

Può capitare che sia necessario utilizzare cicli che non terminano mai (si pensi, ad esempio, all'ascolto da parte di un server su un determinato socket di dati in ingresso dai rispettivi client elaborandoli man mano che questi sono ricevuti).Di seguito pertanto elenchiamo i modi attraverso i quali è possibile inizializzare cicli infiniti in tutti e tre i casi previsti dal linguaggio C con le seguenti:

DEFINIZIONE 15

CICLO FOR INFINITO

```
for (;;) {
        < istruzioni >
}
```

DEFINIZIONE 16

CICLO WHILE INFINITO

```
while ( 1 ) {
        < istruzioni >
}
```

DEFINIZIONE 17

CICLO DO-WHILE INFINITO

```
do {
        < istruzioni >
} while ( 1 );
}
```

Come il lettore immaginerà, mai come in questo caso è necessario ricordare che l'utilizzo di un ciclo rispetto un altro è del tutto irrilevante pur considerando che, di solito, in questi contesti è d'abitudine utilizzare il costrutto while (1). Le ragioni sono, torniamo a dire, del tutto "stilistiche". Capita spesso di dover interrompere un ciclo prima della sua naturale conclusione. In questo caso l'utilizzo della keyword break ci viene incontro, permettendoci di terminarlo in qualunque momento.È opportuno inoltre ricordare che nel caso di cicli annidati utilizzo dell'istruzione break interrompe solo il ciclo più interno lasciando quelli esterni operativi. Questo è particolarmente utile quando si sviluppano applicativi per i quali è necessario prevedere molteplici eccezioni. Nel caso in cui invece vogliamo semplicemente non eseguire alcuna istruzione in presenza di una determinata eccezione utilizzeremo la keyword continue, riportando l'esecuzione all'inizio del ciclo. Come abbiamo visto nella prima parte del corso, un programma scritto in C è un insieme di più semplici sottoprogrammi che, usati congiuntamente, consentono di realizzare operazioni anche molto complesse. In quest'ottica ci riferivamo alla scomposizione di un problema in più operazioni atomiche capaci di risolvere quest'ultimo. In questa sede pertanto daremo una descrizione formale degli argomenti accennati nella prima parte andando ad introdurre uno tra i più importanti concetti del corso: le funzioni. Questo ci permetterà, inoltre, di scrivere il nostro primo programma in C che, lo diciamo fin da subito, non sarà il classico e stravisto "hello world". Se pensiamo per un momento al significato del termine "funzione" in matematica non ci risulterà difficile capire che, anche in informatica, la parola non si distacca molto dal suo senso originario. Intenderemo pertanto una funzione ciò che prima abbiamo espresso come sottoprogramma, ognuna costituita da una sezione dichiarativa ed una esecutiva.
In base a quanto detto precedentemente risulta facile intuire che un buon modo di programmare in senso strutturato è costituito dalla scrittura di funzioni piccole ed estremamente "specializzate" per il loro utilizzo. Queste sono da intendersi come parte di quel processo di "scomposizione" atomica sul quale ci siamo soffermati molto durante la precedente trattazione.
Ognuna di queste ottempererà ad un preciso compito (processo) e potrà essere richiamata ogni qualvolta che quest'ultimo è richiesto nel nostro applicativo, evitando la riscrittura di ulteriore e replicato codice. Diamo pertanto la seguente:

DEFINIZIONE 18

Definiamo ed indichiamo come "funzione" o "sottoprogramma" un insieme di istruzioni definito dal programmatore e non eseguito autonomamente ma soltanto su richiesta da parte del "chiamante". Una funzione è realizzata allo scopo di scomporre il programma in sottoprogrammi componenti capaci di assolvere a ben determinate richieste. Intenderemo pertanto sottoprogramma un meccanismo di astrazione funzionale particolarmente utile nella progettazione del software.
L'esecuzione di un sottoprogramma B è determinata da un'istruzione del programma chiamante A detta "istruzione di chiamata". Il ritorno ad A avviene per mezzo della terminazione delle operazioni effettuate da B ("istruzione di ritorno"). Una funzione lavora su valori in ingresso, interni ed in uscita.
Per valori in ingresso intendiamo quelli già disponibili all'interno di A e trattati dalle operazioni definite in B.
Da qui, intenderemo in uscita quelli calcolati dall'algoritmo definito in B.
I valori interni, invece, si denotano come" variabili interne" e rappresentano quelle utilizzate da B sottoforma di "variabili intermedie o di algoritmo".
Lo scambio di dati tra A e B avviene attraverso il passaggio di parametri formali. La forma sintattica usata per dichiarare funzioni in C è pertanto: < tipo restituito > < nome funzione > (< parametri >).

In C l'unica funzione che deve essere necessariamente presente è la main(), quella alla quale è affidato il controllo dell'applicativo ed il richiamo a tutte le funzioni quando questo è eseguito dal Calcolatore.
Prima di poter essere utilizzata, una funzione va dichiarata. I modi attraverso i quali effettuare la dichiarazione sono sostanzialmente tre: definirla prima della main(), indicarne un prototipo oppure definirla all'interno di una libreria richiamata attraverso il rispettivo header (identificato dalla terminazione .h e contenente i prototipi

delle funzioni implementate all'interno della libreria stessa) nel programma attraverso la direttiva al preprocessore #include. In alcuni compilatori moderni, in particolar modo quelli C++, è altresì possibile invocare funzioni dichiarate in seguito invece che in precedenza, oppure senza prototipo, in quanto l'algoritmo di compilazione crea ricorsivamente il grafo dell'algoritmo invece di effettuare la compilazione in modo sequenziale. Ma è fortemente consigliato invocare una funzione solo dopo che è stata dichiarata, o dopo che è stato inserito il prototipo. Escludendo immediatamente il primo caso per ovvii motivi stilistici/funzionali e lasciando spazio al terzo nelle trattazioni finali del corso diamo la seguente:

DEFINIZIONE 19

Definiamo e denotiamo prototipo di funzione una descrizione formale dello standard ANSI C per riconoscere il tipo del valore di ritorno e dei parametri di una funzione e, conseguentemente, dichiararla prima della sua effettiva definizione ed implementazione. Quest'ultima, definita il prototipo, potrà essere collocata anche dopo la main().
Un prototipo dichiara pertanto il nome della funzione, il tipo del valore restituito dalla stessa (valore di ritorno, void se nullo) ed il tipo dei parametri in ingresso (parametri formali). La forma sintattica per realizzare un prototipo è: "< tipo restituito > < nome funzione > (< tipo parametri in ingresso >);". La collocazione deve essere, ovviamente, antecedente alla main().

IL PRIMO PROGRAMMA

Essendo ormai noti al lettore alcuni concetti elementari per lo sviluppo di applicativi in C possiamo, finalmente, dar luogo al primo approccio operativo del corso con la scrittura di un semplice applicativo che racchiuda alcuni concetti estrinsecati durante queste trattazioni ed alcuni impliciti alla programmazione di tipo Unix-like. In riferimento a questi ultimi è di fondamentale importanza offrire il richiamo teorico 3 (a lato).
Alcune istruzioni risulteranno poco chiare e saranno riprese durante la trattazione dei puntatori nella terza e quarta parte del corso; per ora accettatele così come sono. Di seguito, pertanto, il nostro primo applicativo in C che commenteremo riga per riga.

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <signal.h>
3. #include <string.h>
5. void bof_grep(int);
6.
7. int main()
8. {
9.          signal (SIGSEGV,  bof_grep);
10.         char buffer[10] = {0};
11.         int i;
12.         for(i=0; i<100; i++){
13.                 strcat(buffer, "A");
14.                 fprintf(stdout, "%s\n", buffer);
15.                 }
16.  return 0;
17.  }
18. void bof_grep(int signal)
19. {
20.                 fprintf(stdout, "SIGSEGV
(%d) intercettata. Go away, stack's under
attack!", signal);
21.                 exit(signal);
22. }
```

Alla righe 1-5 sono presenti le apposite direttive al preprocessore relative all'inclusione degli header delle librerie utilizzate nel programma (Standard Input/Output , Standard Library, Signal e String). Alla riga 5 è dichiarata la funzione bof_grep() attraverso il suo prototipo; questa si comporterà da handler per il segnale SIGSEGV (Segment Violation), riceverà pertanto in ingresso il segnale e non restituirà nulla (void). Dalla riga 7 alla 17 è definita la main(). Alla riga 9 viene intercettato attraverso la funzione signal() un segfault tramite il rispettivo Segnale e smistato all'handler (bof_grep()) da noi implementato. Alla riga 10 è dichiarato ed inizializzato un array di 10 caratteri.

Alla riga 11 troviamo la dichiarazione del contatore per il ciclo definito alle righe 12-15 per iterare 100 volte. Alla riga 13, utilizzando la funzione strcat(), concateniamo la lettera "A" all'array definito prima (buffer). Alla riga 14 stampiamo sullo Standard Output (schermo) il valore dell'array attraverso la funzione fprintf(). Alla riga 16 segnaliamo il termine del programma all'OS restituendo il valore 0 attraverso l'apposita keyword del linguaggio "return" (quest'istruzione non sarà mai raggiunta in quanto il programma genererà inevitabilmente un buffer overflow vedi nr. 195). Dalla riga 18 alla 22 definiamo la funzione bof_grep() specificando alle righe 20 e 21 la stampa a video del segnale ricevuto e la terminazione dell'applicativo con il valore restituito dal Segnale con la funzione exit(). Scritto il sorgente con il nostro editor preferito lo compileremo con gcc. Da shell pertanto, supponendo che il sorgente appena scritto si chiami "bof_handler.c", digiteremo:

```
$ gcc bof_handler.c -o bof_handler
```

Come previsto il programma andrà in Segfault ed il Segnale POSIX sarà correttamente intercettato e gestito dall'handler:

```
$ ./bof_handler
A
AA
AAA
...
SIGSEGV (11) intercettata. Go away,
stack's under attack!
```

RICHIAMO TEORICO 3

POSIX SIGNALS.

Con il termine POSIX ci riferiamo ad un insieme di standard (IEEE 1003 - ISO/IEC 9945) sviluppati per assicurare una certa interoperabilità tra diversi OS di tipo Unix. Esistono distinte implementazioni ed espansioni dello standard, una di queste (LSB, Linux Standard Base) è quella adoperata dal sistema operativo GNU/Linux. In linea di massima è possibile sintetizzare lo standard POSIX come quel set di "regole" che un generico Sistema Operativo deve rispettare per essere considerato di tipo UNIX. Tra queste, i Segnali ricoprono un ruolo decisamente fondamentale occupandosi di fornire una metodologia descrittiva concisa attraverso la quale consentire la comunicazione tra processi (IPC - Inter-process communication) su sistemi Unix-like relativa agli eventi occorsi durante l'esecuzione dei processi stessi. L'invio del Segnale è asincrono e si riassume nella notifica al processo dell'evento occorso: la normale esecuzione di quest'ultimo è quindi interrotta dal sistema operativo. È ad esempio un segnale (SIGINT, Keyboard Interrupt Signal, Segnale di Interruzione da tastiera) quello generato premendo contemporaneamente i tasti CTRL e C durante l'esecuzione di un software su OS Unix-like così come un segnale è lanciato nel caso si effettuino divisioni per zero (SIGFPE, Floating Point Exception, Eccezione in Virgola Mobile). Trovate una tabella contenente tutti i POSIX Signals tra gli allegati online di HJ.

Giovanni Federico - giovanni.federico@isek.it
Fabio `BlackLight` Manganiello - blacklight86@gmail.com

PARTE II

CORSO DI PROGRAMMAZIONE IN C

LINGUAGGI Esaurita la necessaria introduzione teorica affrontata nella prima parte del Corso, illustreremo in queste pagine alcuni dei concetti più importanti del linguaggio che, insieme a quanto visto finora, permetteranno di realizzare il nostro primo applicativo in C.

ALGORITMI

Obiettivo principale di qualsiasi percorso elaborativo è definire con precisione una successione di eventi volta alla risoluzione di determinati algoritmi. Lo stesso termine algoritmo è intrinsecamente legato ai medesimi concetti di successione sequenziale o ricorsiva di azioni.

DEFINIZIONE 7

Definiamo algoritmo un procedimento descritto da una sequenza di azioni atte alla risoluzione di un problema assegnato e caratterizzato da inscindibili presupposti di finitezza, descrivibilità, comprensibilità e riproducibilità. La risoluzione di un algoritmo da parte del Calcolatore si traduce nell'esecuzione di una successione (sequenziale e/o ricorsiva) di azioni, ciascuna delle quali con un inizio ed una fine ben determinata e calcolabile.

In merito alla complessità ed alla calcolabilità degli algoritmi risulta, inoltre, necessario offrire la seguente:

DEFINIZIONE 8

Per studiare la trattabilità e la calcolabilità di algoritmi esistono due tipi di complessità computazionale a cui far riferimento: spaziale (quantità di memoria richiesta per immagazzinare i dati necessari) e temporale (tempo richiesto per l'esecuzione dell'algoritmo e la finalizzazione del risultato, spesso identificato anche con il termine "efficienza").

Ai tempi d'oggi la complessità spaziale non costituisce, come un tempo, un parametro di rilievo in quanto i moderni elaboratori si avvalgono, nella stragrande maggioranza dei casi, dell'utilizzo di grossi quantitativi di memoria.
Tuttavia per alcune applicazioni, come gli algoritmi usati in contesti di intelligenza artificiale, la complessità spaziale può diventare un parametro critico, in quanto in tali contesti ricorsivi la complessità spaziale può facilmente crescere con andamento esponenziale al crescere della dimensione del problema.
Ad esempio, le richieste in termini di memoria occupata da un algoritmo intelligente per il gioco degli scacchi possono aumentare di un fattore 10 se aggiungiamo solo una riga e una colonna alla nostra scacchiera. In contesti di tale complessità è facile intuire che lo spazio occupato e l'efficienza dell'algoritmo sono parametri strettamente legati, in quanto i tempi richiesti per l'allocazione e la deallocazione di tanta memoria impattano in modo sensibile sul tempo complessivo di esecuzione dell'algoritmo.
L'efficienza è invece un parametro di assoluta importanza che va considerato in riferimento al contesto nel quale il problema viene risolto. Efficienza, pertanto, non significa considerare il dato temporale in senso assoluto ma collocare quest'ultimo nell'ambiente circostante.
Un algoritmo risulta trattabile se offre risposte in tempi consoni al contesto di riferimento. Sarà invece più efficiente se, negli stessi contesti, offrirà tempi di esecuzione e finalizzazione del risultato minori rispetto ad ulteriori

algoritmi trattabili.
Rispetto a quanto appena detto diamo quindi la seguente:

DEFINIZIONE 9

Identifichiamo con f(n) il tempo di esecuzione di un processo elaborativo, dove n è l'insieme dei dati di input.

Diremo pertanto che un algoritmo gode, ad esempio, di una complessità temporale n3 se il tempo speso per arrivare al risultato dell'elaborazione è pari al cubo della dimensione dell'input (n). Chiarito questo, possiamo offrire la seguente, più generale:

DEFINIZIONE 10

Definiamo e denotiamo come O(f(n)) la complessità temporale di un algoritmo il cui limite superiore del numero di operazioni spese dallo stesso con un input "n" di dati sia f(n), da cui: $O(f(n)) = \{ g(n): \exists\ c, n_0 \mid 0 < g(n) < cf(n)\ \forall\ n > n_0 \}$.

Da qui snodiamo tre tipologie di analisi tese ad individuare quelle operazioni che rappresentino il procedimento dominante dell'intero algoritmo considerato. Avremo pertanto:
- L'analisi del caso migliore in cui la nostra preoccupazione sarà quella di identificare il tempo di esecuzione dell'algoritmo quando "n" rappresenta un insieme di dati di facile trattamento (dimensioni discrete). In termini ingegneristici, nella quasi totalità dei casi, questo tipo di analisi non fornisce nulla di significativamente valido.
-L'analisi del caso medio dove la complessità dei dati da trattare è di difficoltà intermedia.
-L'analisi del caso peggiore quando si presentano difficoltà elevate nel trattamento dei dati. Questo tipo di analisi risulta molto utile nella definizione del tempo massimo che l'algoritmo può impiegare. È importante precisare che non risulta di particolare interesse la dimensione dei dati trattati dal processo bensì come questi crescono all'aumentare degli stessi. In altre parole, particolare attenzione sarà rivolta all'ordine di grandezza di f(n) nella condizione limite. In termini matematici, quando questa tenderà ai suoi asintoti per input sufficientemente grandi. L'analisi asintotica della complessità di un algoritmo consente di restringere ulteriormente il campo ai procedimenti predominanti prima menzionati. Facciamo un esempio: Consideriamo due algoritmi finalizzati alla risoluzione di uno stesso problema con due distinte complessità temporali h(n) e j(n). Nel caso in cui risulti h(n) asintoticamente minore rispetto a j(n) esisterà una "n" (dimensione dell'input) oltre cui l'ordine di grandezza del tempo di esecuzione del primo algoritmo [h(n)] è inferiore rispetto al secondo [j(n)]. Concludiamo riportando a titolo esemplificativo una tabella contenente le dimensioni dell'input e la complessità di tipo esponenziale su un elaboratore capace di gestire un milione di istruzioni al secondo (1 MIPS):

	n = 10	n = 20	n = 30	n = 50
n	0,00001 sec.	0,00002 sec.	0,00003 sec.	0,00005 sec.
n^2	0,0001 sec	0,0004 sec.	0,0009 sec.	0,00025 sec.
n^3	0,001 sec.	0,008 sec	0,027 sec.	0,125 sec.

COSTRUTTI DI CONTROLLO

I concetti legati ad un ben determinato modo di identificare e definire successioni per la risoluzione di algoritmi in un qualsiasi linguaggio di programmazione sono raggruppati in tre distinti modi di affrontare il problema.
È infatti possibile utilizzare un blocco di istruzioni scritto come una sequenza di precise operazioni, singole strutture selettive e costrutti iterativi.
Da questo offriamo pertanto il seguente:

TEOREMA 1

DI BÖHM-JACOPINI - 1966
Algoritmi di qualsiasi complessità possono essere espressi utilizzando indistintamente una singola sequenza di istruzioni, una singola struttura selettiva oppure una singola struttura iterativa.

Definire le azioni elaborative del Calcolatore in C significa adoperare determinati statement (enunciati) che, a seconda dell'impostazione che si vuole dare alla risoluzione degli AR, possono essere dei tipi che vedremo di seguito.

COSTRUTTI SELETTIVI

Come il nome stesso suggerisce, gli enunciati selettivi consentono la definizione di azioni elaborative in base al valore di una determinata espressione booleana. In altri termini si valuta l'espressione di riferimento ed in base al risultato di quest'ultima (true o false) si sceglie se eseguire o meno una istruzione od un blocco di istruzioni.
In questa categoria rientrano gli enunciati "if-else" e "switch". Il primo è un controllo condizionale che consente di valutare l'espressione e, se vera, eseguire un determinato blocco di istruzioni, altrimenti quanto delimitato dall'else. Qualora non fosse definito quest'ultimo statement, l'unico riferimento sarà l'espressione riferita dall'if. In questo caso, se risultasse falsa, nessuna istruzione sarebbe eseguita. Vediamo un rapido esempio:

```
int a, b;
a = b = 20;
if (a == b){ fprintf(stdout, "a e b sono uguali");}
```

Nell'esempio proposto è effettuato un confronto tra due variabili di tipo intero. Nel caso in cui risultino uguali sarà stampato a video l'apposito messaggio. Volendo contemplare anche l'ipotesi in cui i valori non siano uguali, semplicemente, inseriremo un blocco else come di seguito:

```
if (a == b){ fprintf(stdout, "a e b sono uguali");}
else { fprintf(stdout, "a e b non sono uguali"); }
```

Esiste un ulteriore modo di effettuare un confronto selettivo di tipo if-else in C chiamato "if in linea" (o operatore ternario) la cui forma è (controllo) ? <espressione 1 > : <espressione 2>. Tra parentesi tonde indicheremo l'espressione che dovrà essere oggetto di analisi: se questa risulterà vera sarà restituito quanto definito in <espressione 1>, in caso contrario, <espressione 2>. L'esempio seguente chiarirà maggiormente il concetto:

```
int a, b, c;
a = 5; b = 10;
c = (a > b) ? a : b;
fprintf(stdout, "il valore più grande è %d", c);
```

È naturalmente possibile annidare più blocchi if-else:

```
int a, b;
a = 3; b = -4;
if (a > b){
        if (a > 0){ fprintf(stdout, "a è maggiore di b ed è un numero positivo"); }
        else { fprintf(stdout, "a è maggiore di b"); }
} else { fprintf(stdout, "b è' maggiore di a"); }
```

Malgrado fin qui le abbiamo utilizzate, ricordiamo che è possibile omettere le parentesi graffe nel caso in cui si debba eseguire una singola istruzione piuttosto che un blocco. Per effettuare scelte multiple è possibile concatenare più blocchi if-else al fine di contemplare più opzioni: nell'esempio di seguito una semplice implementazione:

```
int a = 1;

if (a == 1)
        fprintf(stdout, "Prima opzione");
else if (a == 2)
        fprintf(stdout, "Seconda opzione");
else if (a == 3)
        fprintf(stdout, "Terza opzione");
else
        fprintf(stdout, "Opzione non contemplata");
```

In alternativa possiamo utilizzare lo statement su scritto (switch) che ben si presta a questi utilizzi e rappresenta un ulteriore modo di eseguire vari blocchi di istruzioni in funzione del valore dell'espressione definita. L'esempio appena proposto, utilizzando lo switch, diventa:

```
int a = 3;
switch(a){
        case 1:
                fprintf(stdout, "Prima opzione");
                break;
        case 2:
                fprintf(stdout, "Seconda opzione");
                break;
        case 3:
                fprintf(stdout, "Terza opzione");
                break;
        default:
                fprintf(stdout, "Opzione non contemplata");
}
```

L'istruzione "break" ci consente di interrompere l'esecuzione dello switch una volta eseguito il singolo blocco di istruzioni definito da una delle eventualità previste (case). Incontreremo di nuovo questa keyword nel prosieguo di questa trattazione, non appena avremo necessità di uscire da un ciclo. Concludiamo pertanto definendo formalmente la sintassi del costrutto "if-else" nella sua forma classica:

DEFINIZIONE 11

STATEMENT IF-ELSE

```
if (espressione di controllo)  {
        < blocco di istruzioni nel caso in cui l'espressione di controllo risulti verificata (TRUE) >;
}
else {
        < blocco di istruzioni nel caso in cui l'espressione di controllo non risulti verificata (FALSE) >;
}
```

COSTRUTTI ITERATIVI

Gli statement iterativi consentono di eseguire un determinato blocco di istruzioni in modo reiterato oppure fintantoché una determinata espressione risulta vera. In questo caso, stabilendo un'opportuna "condizione", la ripetizione delle istruzioni terminerà solo quando questa risulterà soddisfatta. I cicli costituiscono un'entità fondamentale della programmazione strutturata e consentono di esprimere al meglio le capacità di elaborazione della macchina. Permettendo di eseguire più volte un determinato blocco di istruzioni rendono particolarmente agevole l'implementazione su calcolatore di algoritmi anche molto complessi. È importante segnalare fin da ora che nella totalità dei casi l'utilizzo di un tipo di ciclo piuttosto che un altro non deve intendersi come impossibilità da parte dell'uno o dell'altro di soddisfare le operazioni richieste, bensì di ottimizzare e spesso rendere più leggibile l'intero sorgente. Il linguaggio C dispone di vari cicli, di seguito li analizzeremo tutti offrendo per ognuno di essi qualche esempio. Questo ci permetterà di capire meglio gli utilizzi e le modalità di esecuzione degli stessi.

IL CICLO FOR

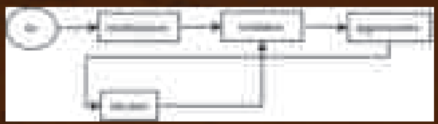
Il primo ciclo oggetto della nostra trattazione sarà quello di tipo enumerativo o ciclico ovvero quello da preferirsi ogni qualvolta che il numero di ripetizioni dei blocchi d'operazioni da effettuare è noto a priori.
La keyword del linguaggio preposta alla definizione dello stesso è "for".
L'inizializzazione del ciclo è effettuata mediante l'utilizzo di uno statement composto da un'istruzione di assegnazione attraverso la quale viene fornito il valore iniziale alla variabile che fungerà da contatore del ciclo, una condizione rappresentante un'espressione booleana atta ad identificare la terminazione ed, infine, un'apposita istruzione che definirà come la variabile utilizzata come contatore debba cambiare il suo valore ad ogni ripetizione.
Nella quasi totalità dei casi quest'ultima istruzione rappresenta un incremento o un decremento del valore della variabile contatore. A tal fine si utilizzano gli operatori di post e pre incremento/decremento trovati come allegati alla prima parte di questo corso nell'apposita tabella disponibile sul sito della rivista in formato pdf.
In base a quanto finora scritto possiamo pertanto offrire la seguente:

DEFINIZIONE 12

IL CICLO FOR

```
for ( < contatore >; < condizione >; < aggiornamento contatore > ) {
        < istruzioni >
}
```

Nell'immagine di seguito riportata è schematizzato il funzionamento del ciclo for.



Il funzionamento del ciclo for.

Un tipico esempio di applicazione del ciclo for è riscontrabile nel calcolo del

fattoriale di un numero naturale come notiamo nel seguente esempio:

```
int n = 10; // Il numero per il quale si intende
effettuare il fattoriale
int fattoriale = 1; // Inizializzazione della var.
che conterrà il fattoriale
int contatore; // Il contatore per il ciclo for

for(contatore = n; contatore > 1; contatore--){
        fattoriale *= contatore;
}

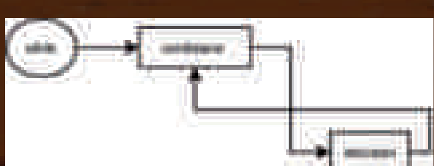
fprintf(stdout, "Il fattoriale del numero %d è
%d.", n, fattoriale);
```

IL CICLO WHILE

Il ciclo while è il tipo di interazione più semplice del linguaggio ed è composto da un blocco di istruzioni che vengono eseguite ciclicamente fintanto che l'espressione di controllo risulta verificata.

Questo aspetto è di primaria importanza perché l'esecuzione del while è subordinata alla valutazione di quest'ultima espressione. L'interruzione del ciclo, inoltre, si verificherà quando la medesima assumerà il valore booleano FALSE (ovvero, non risulterà più verificata).

Da questo risulta facile immaginare che il ciclo può terminare solo se all'interno del blocco di istruzioni referenziato da esso siano presenti statement che modifichino il valore di verità dell'espressione di controllo. Qualora questo non accadesse il ciclo non potrebbe ovviamente concludersi dando luogo ad imbarazzanti errori durante l'esecuzione del nostro applicativo.



Il funzionamento del ciclo while.

Spese queste poche parole introduttive possiamo pertanto dare la seguente:

DEFINIZIONE 13

IL CICLO WHILE

```
while ( < condizione > ) {
        < istruzioni >
}
```

Come abbiamo detto poc'anzi la scelta di un ciclo piuttosto che un altro per la risoluzione di un determinato algoritmo non avviene perché l'uno può fare ciò che l'altro non può. In virtù di ciò, a titolo esemplificativo, offriamo il calcolo del fattoriale di un numero utilizzando, piuttosto che il for, il ciclo while.
Il lettore noterà autonomamente l'incremento di righe di codice necessario per esprimere lo stesso algoritmo con un ciclo diverso da quello che meglio si presta a questi scopi (in questo contesto while piuttosto che for).
Da qui, di nuovo, il richiamo relativo alla scelta del ciclo migliore rispetto agli utilizzi che se ne devono fare.

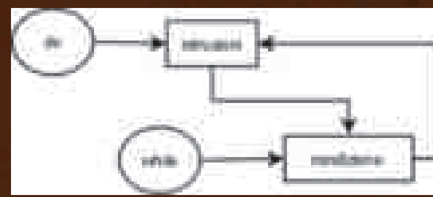
```
int n = 10;
int fattoriale = 1;
int contatore = 0;

while (contatore != n) {
        fattoriale *= (n - contatore);
        contatore++;
}

fprintf(stdout, "Il fattoriale del numero %d è
%d.", n, fattoriale);
```

IL CICLO DO-WHILE

Vi sono determinate situazioni in cui è necessario eseguire un blocco di istruzioni all'interno di un ciclo indipendentemente dal valore dell'espressione di controllo. In altri termini esistono scenari all'interno dei quali è necessario eseguire almeno una volta un ciclo.
In questi casi risulta particolarmente utile avvalersi del costrutto do-while che, a differenza del while, verifica l'espressione condizionale alla fine. Il blocco istruzioni viene quindi prima eseguito e poi valutato.
A questo punto, il blocco sarà ancora eseguito finché la condizione risulterà verificata.



Il funzionamento del ciclo do-while.

Diamo pertanto la seguente:

DEFINIZIONE 14

IL CICLO DO-WHILE

```
do {
        < istruzioni >
} while ( < condizione > );
```

Per le stesse motivazioni viste nel while, offriamo ancora una volta lo stesso esempio del fattoriale di un numero naturale, stavolta con il ciclo do-while:

```
int n = 10;
int fattoriale = 1;
int contatore = 0;

do {
        fattoriale *= (n - contatore);
        contatore++;
} while (contatore != n);

fprintf(stdout, "Il fattoriale del numero %d è
%d.", n, fattoriale);
```

CICLI NESTED

Come per i costruttivi selettivi è naturalmente possibile annidare cicli all'interno di altri cicli. Di seguito riportiamo un esempio con un ciclo for la cui logica tornerà utile nelle prossime trattazioni relative alla manipolazione delle matrici. Precisiamo inoltre che quanto di seguito descritto è applicabile con qualunque ciclo supportato dal linguaggio.

```
int i, j, k;
k = 1;

for(i = 0; i < 10; i++){
    for(j = 0; j < 10; j++) {

        if(k < 10)
            fprintf(stdout, "00%d ", k);
        else if (k < 100)
```

```
                fprintf(stdout, "0%d ", k);
            else
                fprintf(stdout, "%d ", k);
        k++;
        }
        fprintf(stdout, "\n");
}
```

Risultato di questa elaborazione sarà la stampa a video dei primi 100 numeri ordinati in una tabella 10 x 10 (vedi immagine).



Tabella 10 x 10 creata con due for annidati.

LOOP

Può capitare che sia necessario utilizzare cicli che non terminano mai (si pensi, ad esempio, all'ascolto da parte di un server su un determinato socket di dati in ingresso dai rispettivi client elaborandoli man mano che questi sono ricevuti).Di seguito pertanto elenchiamo i modi attraverso i quali è possibile inizializzare cicli infiniti in tutti e tre i casi previsti dal linguaggio C con le seguenti:

DEFINIZIONE 15

CICLO FOR INFINITO

```
for (;;) {
        < istruzioni >
}
```

DEFINIZIONE 16

CICLO WHILE INFINITO

```
while ( 1 ) {
        < istruzioni >
}
```

DEFINIZIONE 17

CICLO DO-WHILE INFINITO

```
do {
        < istruzioni >
} while ( 1 );
}
```

Come il lettore immaginerà, mai come in questo caso è necessario ricordare che l'utilizzo di un ciclo rispetto un altro è del tutto irrilevante pur considerando che, di solito, in questi contesti è d'abitudine utilizzare il costrutto while (1). Le ragioni sono, torniamo a dire, del tutto "stilistiche". Capita spesso di dover interrompere un ciclo prima della sua naturale conclusione. In questo caso l'utilizzo della keyword break ci viene incontro, permettendoci di terminarlo in qualunque momento.È opportuno inoltre ricordare che nel caso di cicli annidati utilizzo dell'istruzione break interrompe solo il ciclo più interno lasciando quelli esterni operativi. Questo è particolarmente utile quando si sviluppano applicativi per i quali è necessario prevedere molteplici eccezioni. Nel caso in cui invece vogliamo semplicemente non eseguire alcuna istruzione in presenza di una determinata eccezione utilizzeremo la keyword continue, riportando l'esecuzione all'inizio del ciclo. Come abbiamo visto nella prima parte del corso, un programma scritto in C è un insieme di più semplici sottoprogrammi che, usati congiuntamente, consentono di realizzare operazioni anche molto complesse. In quest'ottica ci riferivamo alla scomposizione di un problema in più operazioni atomiche capaci di risolvere quest'ultimo. In questa sede pertanto daremo una descrizione formale degli argomenti accennati nella prima parte andando ad introdurre uno tra i più importanti concetti del corso: le funzioni. Questo ci permetterà, inoltre, di scrivere il nostro primo programma in C che, lo diciamo fin da subito, non sarà il classico e stravisto "hello world". Se pensiamo per un momento al significato del termine "funzione" in matematica non ci risulterà difficile capire che, anche in informatica, la parola non si distacca molto dal suo senso originario.
Intenderemo pertanto una funzione ciò che prima abbiamo espresso come sottoprogramma, ognuna costituita da una sezione dichiarativa ed una esecutiva.
In base a quanto detto precedentemente risulta facile intuire che un buon modo di programmare in senso strutturato è costituito dalla scrittura di funzioni piccole ed estremamente "specializzate" per il loro utilizzo. Queste sono da intendersi come parte di quel processo di "scomposizione" atomica sul quale ci siamo soffermati molto durante la precedente trattazione.
Ognuna di queste ottempererà ad un preciso compito (processo) e potrà essere richiamata ogni qualvolta che quest'ultimo è richiesto nel nostro applicativo, evitando la riscrittura di ulteriore e replicato codice.
Diamo pertanto la seguente:

DEFINIZIONE 18

Definiamo ed indichiamo come "funzione" o "sottoprogramma" un insieme di istruzioni definito dal programmatore e non eseguito autonomamente ma soltanto su richiesta da parte del "chiamante". Una funzione è realizzata allo scopo di scomporre il programma in sottoprogrammi componenti capaci di assolvere a ben determinate richieste. Intenderemo pertanto sottoprogramma un meccanismo di astrazione funzionale particolarmente utile nella progettazione del software. L'esecuzione di un sottoprogramma B è determinata da un'istruzione del programma chiamante A detta "istruzione di chiamata". Il ritorno ad A avviene per mezzo della terminazione delle operazioni effettuate da B ("istruzione di ritorno"). Una funzione lavora su valori in ingresso, interni ed in uscita. Per valori in ingresso intendiamo quelli già disponibili all'interno di A e trattati dalle operazioni definite in B. Da qui, intenderemo in uscita quelli calcolati dall'algoritmo definito in B. I valori interni, invece, si denotano come" variabili interne" e rappresentano quelle utilizzate da B sottoforma di "variabili intermedie o di algoritmo". Lo scambio di dati tra A e B avviene attraverso il passaggio di parametri formali. La forma sintattica usata per dichiarare funzioni in C è pertanto: < tipo restituito > < nome funzione > (< parametri >).

In C l'unica funzione che deve essere necessariamente presente è la main(), quella alla quale è affidato il controllo dell'applicativo ed il richiamo a tutte le funzioni quando questo è eseguito dal Calcolatore.
Prima di poter essere utilizzata, una funzione va dichiarata. I modi attraverso i quali effettuare la dichiarazione sono sostanzialmente tre: definirla prima della main(), indicarne un prototipo oppure definirla all'interno di una libreria richiamata attraverso il rispettivo header (identificato dalla terminazione .h e contenente i prototipi

delle funzioni implementate all'interno della libreria stessa) nel programma attraverso la direttiva al preprocessore #include. In alcuni compilatori moderni, in particolar modo quelli C++, è altresì possibile invocare funzioni dichiarate in seguito invece che in precedenza, oppure senza prototipo, in quanto l'algoritmo di compilazione crea ricorsivamente il grafo dell'algoritmo invece di effettuare la compilazione in modo sequenziale. Ma è fortemente consigliato invocare una funzione solo dopo che è stata dichiarata, o dopo che è stato inserito il prototipo. Escludendo immediatamente il primo caso per ovvii motivi stilistici/funzionali e lasciando spazio al terzo nelle trattazioni finali del corso diamo la seguente:

DEFINIZIONE 19

Definiamo e denotiamo prototipo di funzione una descrizione formale dello standard ANSI C per riconoscere il tipo del valore di ritorno e dei parametri di una funzione e, conseguentemente, dichiararla prima della sua effettiva definizione ed implementazione. Quest'ultima, definita il prototipo, potrà essere collocata anche dopo la main().
Un prototipo dichiara pertanto il nome della funzione, il tipo del valore restituito dalla stessa (valore di ritorno, void se nullo) ed il tipo dei parametri in ingresso (parametri formali). La forma sintattica per realizzare un prototipo è: "< tipo restituito > < nome funzione > (< tipo parametri in ingresso >);". La collocazione deve essere, ovviamente, antecedente alla main().

IL PRIMO PROGRAMMA

Essendo ormai noti al lettore alcuni concetti elementari per lo sviluppo di applicativi in C possiamo, finalmente, dar luogo al primo approccio operativo del corso con la scrittura di un semplice applicativo che racchiuda alcuni concetti estrinsecati durante queste trattazioni ed alcuni impliciti alla programmazione di tipo Unix-like. In riferimento a questi ultimi è di fondamentale importanza offrire il richiamo teorico 3 (a lato).
Alcune istruzioni risulteranno poco chiare e saranno riprese durante la trattazione dei puntatori nella terza e quarta parte del corso; per ora accettatele così come sono. Di seguito, pertanto, il nostro primo applicativo in C che commenteremo riga per riga.

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <signal.h>
3. #include <string.h>
5. void bof_grep(int);
6.
7. int main()
8. {
9.        signal (SIGSEGV,  bof_grep);
10.       char buffer[10] = {0};
11.       int i;
12.       for(i=0; i<100; i++){
13.              strcat(buffer, "A");
14.              fprintf(stdout, "%s\n", buffer);
15.                 }
16.  return 0;
17.  }
18. void bof_grep(int signal)
19. {
20.              fprintf(stdout, "SIGSEGV (%d) intercettata. Go away, stack's under attack!", signal);
21.              exit(signal);
22. }
```

Alla righe 1-5 sono presenti le apposite direttive al preprocessore relative all'inclusione degli header delle librerie utilizzate nel programma (Standard Input/Output , Standard Library, Signal e String). Alla riga 5 è dichiarata la funzione bof_grep() attraverso il suo prototipo; questa si comporterà da handler per il segnale SIGSEGV (Segment Violation), riceverà pertanto in ingresso il segnale e non restituirà nulla (void). Dalla riga 7 alla 17 è definita la main(). Alla riga 9 viene intercettato attraverso la funzione signal() un segfault tramite il rispettivo Segnale e smistato all'handler (bof_grep()) da noi implementato. Alla riga 10 è dichiarato ed inizializzato un array di 10 caratteri.

Alla riga 11 troviamo la dichiarazione del contatore per il ciclo definito alle righe 12-15 per iterare 100 volte. Alla riga 13, utilizzando la funzione strcat(), concateniamo la lettera "A" all'array definito prima (buffer). Alla riga 14 stampiamo sullo Standard Output (schermo) il valore dell'array attraverso la funzione fprintf(). Alla riga 16 segnaliamo il termine del programma all'OS restituendo il valore 0 attraverso l'apposita keyword del linguaggio "return" (quest'istruzione non sarà mai raggiunta in quanto il programma genererà inevitabilmente un buffer overflow vedi nr. 195). Dalla riga 18 alla 22 definiamo la funzione bof_grep() specificando alle righe 20 e 21 la stampa a video del segnale ricevuto e la terminazione dell'applicativo con il valore restituito dal Segnale con la funzione exit(). Scritto il sorgente con il nostro editor preferito lo compileremo con gcc. Da shell pertanto, supponendo che il sorgente appena scritto si chiami "bof_handler.c", digiteremo:

```
$ gcc bof_handler.c -o bof_handler
```

Come previsto il programma andrà in Segfault ed il Segnale POSIX sarà correttamente intercettato e gestito dall'handler:

```
$ ./bof_handler
A
AA
AAA
...
SIGSEGV (11) intercettata. Go away, stack's under attack!
```

RICHIAMO TEORICO 3

POSIX SIGNALS.

Con il termine POSIX ci riferiamo ad un insieme di standard (IEEE 1003 - ISO/IEC 9945) sviluppati per assicurare una certa interoperabilità tra diversi OS di tipo Unix. Esistono distinte implementazioni ed espansioni dello standard, una di queste (LSB, Linux Standard Base) è quella adoperata dal sistema operativo GNU/Linux. In linea di massima è possibile sintetizzare lo standard POSIX come quel set di "regole" che un generico Sistema Operativo deve rispettare per essere considerato di tipo UNIX. Tra queste, i Segnali ricoprono un ruolo decisamente fondamentale occupandosi di fornire una metodologia descrittiva concisa attraverso la quale consentire la comunicazione tra processi (IPC - Inter-process communication) su sistemi Unix-like relativa agli eventi occorsi durante l'esecuzione dei processi stessi. L'invio del Segnale è asincrono e si riassume nella notifica al processo dell'evento occorso: la normale esecuzione di quest'ultimo è quindi interrotta dal sistema operativo. È ad esempio un segnale (SIGINT, Keyboard Interrupt Signal, Segnale di Interruzione da tastiera) quello generato premendo contemporaneamente i tasti CTRL e C durante l'esecuzione di un software su OS Unix-like così come un segnale è lanciato nel caso si effettuino divisioni per zero (SIGFPE, Floating Point Exception, Eccezione in Virgola Mobile). Trovate una tabella contenente tutti i POSIX Signals tra gli allegati online di HJ.

Massimiano Rinaldi
redazione@hackerjournal.it

WI-FI: QUANDO LA BARRIERA È FISICA

WIRELESS

CI SONO MOLTI MODI DI ATTACCARE UNA RETE WI-FI, MA AL DI LÀ DELLE BARRIERE SOFTWARE ERETTE PER DIFENDERSI, CI SONO BARRIERE FISICHE CHE POSSONO ESSERE ALTRETTANTO EFFICACI E INVALICABILI.

Le reti wireless sono una specie di benedizione per gli utenti ma una sorta di maledizione per gli amministratori di sistema.
Abbiamo più volte parlato all'interno delle pagine di Hacker Journal delle più comuni chiavi di protezione della reti senza fili, WEP e WPA, ma esistono delle barriere fisiche che possono circoscrivere il segnale basato su protocollo IEEE 802.11
Attualmente le reti wireless utilizzano una gamma di alta frequenza dello spettro elettromagnetico a 2,4 GHZ o 5 GHZ che permette alla lunghezza d'onda di attraversare oggetti apparentemente solidi.
Ci sono però in natura alcuni ostacoli che riducono la qualità del segnale. I muri di mattoni, ad esempio, per loro natura trattengono molta acqua e questa può bloccare l'energia delle frequenza a 2,4 GHZ, lo stesso vale per il metallo nelle pareti, o per le tubature. Insomma il segnale viaggia nell'aria ma subisce delle limitazioni che possono anche essere convenientemente utilizzate per cercare di circoscrivere il segnale ed evitare che sia troppo percepibile all'esterno dell'edificio, sia esso una casa privata o la sede di un'azienda.

MATERIALI A BASSO DEGRADO DI SEGNALE

Legno, Plastica, Materiali sintetici, Amianto e Vetro non oppongono una grande resistenza al passaggio del segnale. Radio. Il degrado è minimo. In questa categoria possiamo includere finestre, tramezzi d'ufficio, muri interni e altri elementi realizzati proprio con questa tipologia di materiale.

Quindi un'open space separato in uffici con strutture prefabbricate di plastica e vetro non dovrebbe avere grossi problemi alla diffusione del segnale.

MATERIALI MENO PENETRABILI

Acqua, Mattoni e Marmo possono determinare un degrado sensibile, a livello medio, del segnale radio. L'acqua non va intesa in senso stretto, chiaro che se è presente un acquario che occupa gran parte di una stanza questo può offrire una certa resistenza al passaggio delle onde radio, ma, in senso più lato, si intende anche solo la presenza di materiali che in qualche modo trattengono l'acqua. Ad esempio il legno, che da solo non offre una grande resistenza al passaggio del segnale radio, ma il legno umido, quindi impregnato da particelle d'acqua, può diventare un ostacolo serio.
Gli stessi mattoni dei muri esterni presentano una buona propensione a trattenere acqua e proprio questa rappresenta, unitamente al materiale di cui è costituito il mattone, un ostacolo decisamente poco penetrabile.



ALTO DEGRADO DI SEGNALE

Ci sono poi dei materiali che presentano un indice di impenetrabilità alle onde radio decisamente alto e che bisogna tenere in debita considerazione quando si progetta una rete senza fili. La carta costituisce una barriera decisamente isolante. Si parla quindi di giornali, ma anche di grandi cartine appese ai muri, oppure di poster. Anche il cemento armato dei pavimenti e dei muri esterni ha una grande capacità di opporsi al passaggio delle onde radio, così come i vetri antiproiettile.

GLI IMPENETRABILI

C'è infine una categoria, quella dei metalli, che risulta quasi impenetrabile alle onde radio e produce, nella migliore delle ipotesi, un degrado del segnale radio davvero consistente. Si parla in questo caso di scrivanie, tramezzi d'ufficio, ma anche delle armature del cemento armato.

TECNICHE DI DIFESA

Una difesa fisica da intrusioni esterne può essere impostata cercando di collocare il router che trasmette le frequenze radio all'interno dell'ufficio, lontano dai muri perimetrali e soprattutto dalla finestre. Infatti i muri, siano essi in mattoni o cemento armato, così come il pavimento, offrono da soli già una discreta barriera alla potenza del segnale.

UN AIUTINO PLEASE

Ci può essere anche l'ipotesi opposta. Ovvero che ci si debba preoccupare non tanto della possibile diffusione, e quindi intercettazione, del segnale radio all'esterno dell'edificio o del locale, quanto dell'insufficiente potenza del segnale stesso per gli scopi a cui è preposto.

In questi casi i possono utilizzare delle antenne in grado di coadiuvare, come potenza di trasmissione del segnale, quelle fornite di serie col router.

Si parla di antenne omnidirezionali, yagi e paraboliche.
L'antenna ominidirezionale è perfetta per irradiare il segnale in tutte le direzioni.
Le antenne yagi assomigliano ad un piccolo tubo di plastica, hanno un raggio abbastanza focalizzato, e devono essere puntate verso la postazione remota.
Infine, l'antenna parabolica rappresenta il modello più potente in circolazione, esteriormente assomiglia ad un piatto, a volte grigliato dotato di un piccolo satellite centrale.
Viene utilizzata per coprire lunghe distanze.